młody TECHNIK

InforMik

MAGAZYN KOMPUTEROWY „MŁODEGO TECHNIKA"

III
1987



Nasz komentarz:

DOKTRYNERZY I DOGMATYCY

Ostatnio pewien dwudziestoletni entuzjasta komputerów zadał mi pytanie, na jakim sprzęcie i w jakich językach programowania stawiałem pierwsze samodzielne kroki w informatyce. Gdy, zgodnie z prawdą, odpowiedziałem, że na MERZE-60 z pamięcią 24 KB i w językach: BASIC i Fortran (inne nie były dostępne), młody człowiek żachnął się z obrzydzeniem: dla niego pojęcie komputera zaczyna się od IBM PC, zaś jedynymi językami godnymi prawdziwego informatyka są C i Prolog, no, ewentualnie jeszcze Pascal. Ale żeby ten, no, jak mu tam... aha — Fortran (fuj! jakie brzydkie słowo)!? Młodzieniec nie omieszkał wyrazić swego ubolewania z powodu faktu, że świat rekursji, nawrotów, list, kolejek itd. pozostanie dla mnie na zawsze zamknięty. Cóż, zepsułem sobie charakter. Zetknięcie z BASIC-em czy Fortranem to dla programisty gorsze niż trąd, powodując nieodwracalną degenerację.

Kiedy stwierdziłem, że na archaicznej MERZE-60 robiliśmy całkiem fajne rzeczy, ze stosowaniem kolejek i nawrotów radziliśmy sobie nawet w BASIC-u, a znajomość Fortranu nie przeszkadza mi w zrozumieniu istoty rekursji, mój rozmówca poczuł się nieco urażony. Nie mam wątpliwości — nie uwierzył.

Czytając popularną prasę informatyczną co rusz można spotkać artykuły pt. „Dlaczego nie cierpię BASIC-a" itp. Ich autorzy wyszydzają BASIC i straszą nim młodych programistów, jak niegdyś dzieci straszono Babą Jagą. Szkopuł w tym, że języki programowania nie są po to, aby o nich pisać i dywagować, ani po to, aby je lubić lub nie, lecz po to, aby używać ich jako narzędzia. Przyznam, że nie lubię swego podręcznego młotka, ale gdy trzeba szybko wbić gwóźdź w ścianę, chwytam bez wahania jego wysłużony trzonek. To nic, że obok w PEWEX-ie można nabyć wspaniały, elektryczny młotek firmy Black & Decker (panie — sam wbija!). Mój kawał stali na kijku całkiem mi wystarcza w podręcznych pracach, gdyż biegle już opanowałem posługiwanie się nim, znam jego kaprysy i wiem, kiedy sfatygowany obuch może przytłuc mi paluchy.

BASIC spełnia w warsztacie programisty rolę owego młotka do doraźnych prac. Im więcej narzędzi w warsztacie, tym większa szansa znalezienia najodpowiedniejszego do danej pracy. Nawet jeśli nie wszystkimi narzędziami posługujemy się biegle, to jednak znajomość każdego z nich podnosi naszą ogólną wiedzę o fachu, który uprawiamy, rozszerza nasze horyzonty. Warto znać nie tylko narzędzia wydajne, a złożone w użyciu, ale i te proste, podręczne. Nie bez powodu starzy rzemieślnicy szkolenie uczniów zaczynali od pilnika i skrobaka, choć później stawiali ich przy obrabiarce.

W moim przekonaniu, najlepszą rzeczą, jaką może zrobić dziś młody adept informatyki, nie jest bynajmniej szczegółowe zgłębianie jednego (tego najlepszego...) języka programowania, ale raczej troska o własną wszechstronność. Jednym słowem, uczyć się czego tylko się da, otrzeć się o Prolog lub Lisp, ale i otrzeć się o asembler. Sprawność w posługiwaniu się konkretnym językiem-narzędziem przyjdzie szybko i niepostrzeżenie, w chwili, gdy zapadnie decyzja implementacji w nim danego algorytmu.

Trafna analiza problemu i wypływający z niej wybór narzędzia więcej wtedy znaczy, niż biegłość w posługiwaniu się jednym (tym najlepszym...) językiem, który do danego zastosowania pasuje akurat tak, jak kapiszon do haubicy. Poza tym — w każdym języku można programować poprawnie, w każdym też można partaczyć. W jednym języku uzyskanie porządnej struktury programu jest prostsze, w innym — trudniejsze. O ostatecznym efekcie decyduje jednak nie sam język, lecz człowiek i jego umiejętności. Znam osoby, które piszą obszerne, lecz mimo to pięknie strukturowane i przejrzyste programy w BASIC-u oraz takie, które wprawdzie zarzekają się, że nigdy w życiu nie użyły rozkazu GO TO, lecz produkują programy pascalowe zdradzające większe pokrewieństwo z fasolką po bretońsku, niż z naturalnym charakterem rozwiązywanego problemu.

W codziennej praktyce programisty jest sporo drobnych problemów, które najekonomiczniej można rozwiązać właśnie w BASIC-u, a nie np. w TURBO-Pascalu lub w Prologu. Zgadzam się z twierdzeniem, że BASIC nie powinien służyć do elementarnej edukacji informatycznej. Jeśli jednak ktoś decyduje się na bardziej zażyłą znajomość z informatyką, to kontakt z tym językiem nie powinien mu zaszkodzić.

Dogmatycy i doktrynerzy zepsuli już w historii sporo krwi, tak sobie, jak i bliźnim. Bywało tak i w informatyce, jak chociażby dawno temu w zaprzyjaźnionym z nami mocarstwie, gdzie ze względów głównie pozatechnicznych obowiązującym językiem uczyniono Algol, podczas gdy reszta świata posługiwała się Fortranem.

Najlepszą obroną przed doktrynerstwem jest po prostu własne, wyrobione zdanie. Aby je mieć, nie wystarczy jednak biernie przyswajać sobie cudze opinie — trzeba samemu eksperymentować i konfrontować obiegowe opinie z praktyką.

Wbrew obowiązującej dziś w niektórych środowiskach modzie, traktującej posądzenie o posługiwanie się BASIC-em jako ciężką obelgę, apeluję: uczcie się także BASIC-a. BASIC jest tu zresztą tylko pretekstem: przykładów informatycznego doktrynerstwa można by przytoczyć więcej. A swoją drogą, ciekawe, jak programowi przeciwnicy BASIC-a potraktują pojawienie się dialektu TURBO-BASIC? Jakby nie było, lansuje go przecież słynna firma Borland, której zawdzięczamy m.in. TURBO-Pascal i TURBO-Prolog...

Roland Wacławek



młody TECHNIK
InforMik

MAGAZYN KOMPUTEROWY MŁODEGO TECHNIKA
NR III ROCZNIK I

SPIS TREŚCI

OKŁADKA:
I — Widoczna na zdjęciu „szafa" to zbudowana na lampach elektronowych jednostka pamięci 16 kilobajtów pochodząca z lat pięćdziesiątych — dziś eksponat amerykańskiego Muzeum Komputerów w Marlboro w stanie Massachusetts. Obok pod mikroskopem, obejrzeć można współczesną pamięć 16 KB.
III i IV — Mikrokomputer jednoukładowy firmy Intel.

Numer ilustrował: Jerzy Flisak
Zdjęcia w numerze: Władysław P. Jabłoński, Grzegorz Zalot, ze zbiorów redakcji.

Na SZEROKIE TORY

Nie, nie będzie to felieton na tematy kolejowe, choć można by od tego zacząć, gdyż normalizacja jest dziedziną obejmującą dosłownie wszystko — od śrubki do parowozu. Proces ujednolicania norm, tworzenia standardów i unifikacja już od dość dawna wkroczyły w świat techniki i to zarówno tej przez małe, jak i przez duże t.

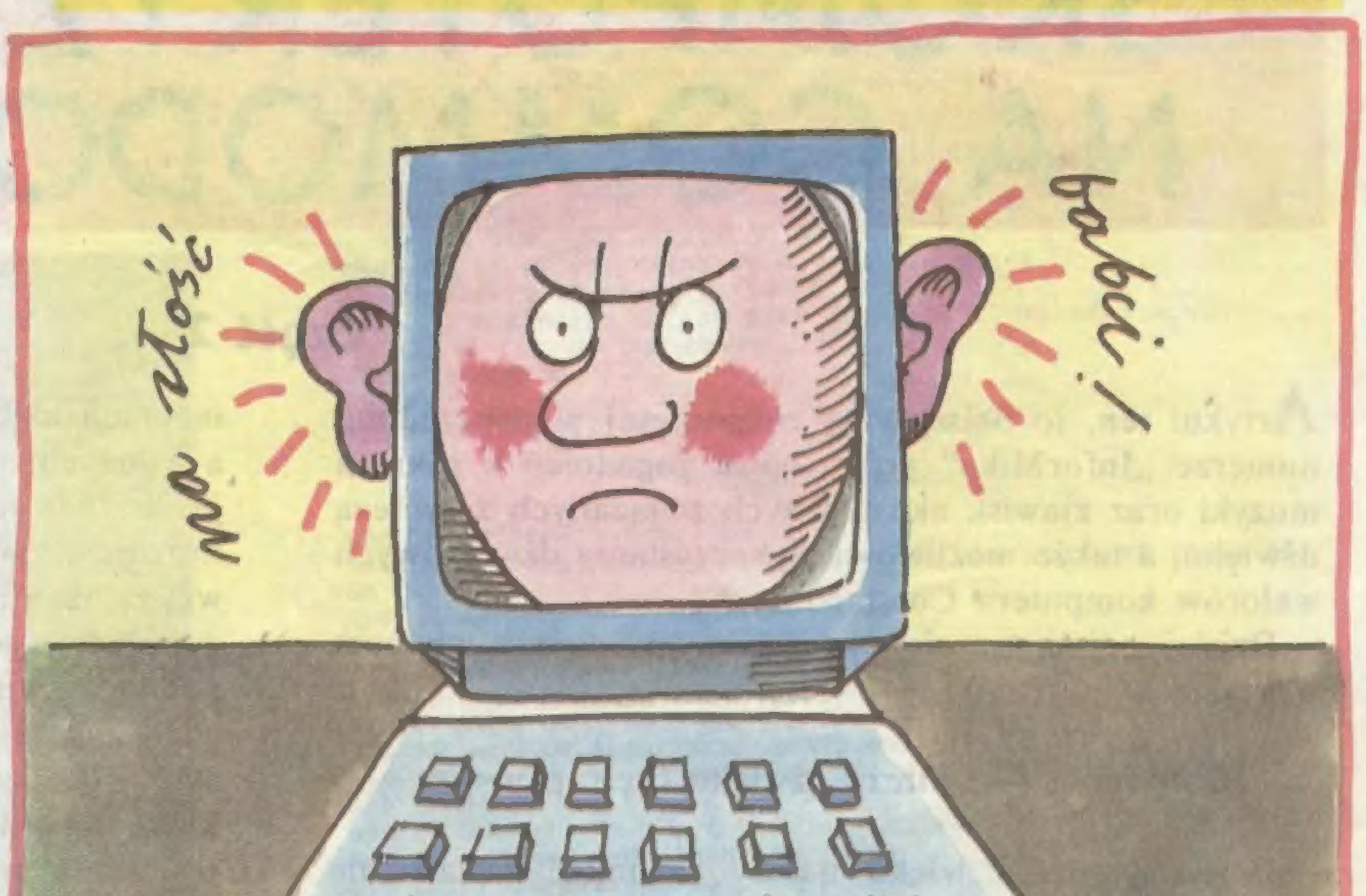
Z pełnym zaakceptowaniem ustalonych norm bywa czasem różnie — są państwa, które uznają tylko własne normy i nie myślą podporządkować się międzynarodowym ustaleniom, w sumie jednak — prędzej czy później — trzeba się pogodzić z układem norm przyjętym przez większość, bo zasada zrobienia babci na złość i odmrożenia sobie uszu na dłuższą metę okazuje się dość bolesna...

Piszę o tym dlatego, że normalizacja wkracza coraz szerzej i zdecydowaniej do informatyki. Próby określania międzynarodowych standardów są podejmowane mimo sprzeciwów ze strony firm komputerowych, w których interesie jest tworzenie sprzętu nietypowego, bo pociąga to za sobą fakt konieczności korzystania ze specjalnego oprzyrządowania i specjalistycznego serwisu, co zawsze wiąże się dla producenta z dodatkowymi zyskami (wiadomo, że nic „obcego" nie będzie do „naszego" komputera pasować).

Mimo rozpaczliwych prób obrony dotychczasowych pozycji w komputerowym chaosie producentów sprzętu i oprogramowania coraz wyraźniej daje się zauważyć, że zwycięstwo chyli się w stronę firm-strategów dążących do osiągnięcia dalekosiężnych celów, a nie tych, którzy pragnęli osiągnąć doraźny, czysto finansowy zysk. Być może kiedyś Księga Rodzaju komputerowej biblii zacznie się od słów: „Na początku był IBM", gdyż ta firma stworzyła właściwie pierwszy standard obowiązujący w świecie mikrokomputerów pomimo wszelkich istniejących w nim różnic.

Człowiek — istota w swej naturze wolna i nie lubiąca ramek, granic czy sztucznych podziałów — niekiedy jest zmuszony do elementarnego choć usystematyzowania świata, w którym żyje. W tym też celu stworzył sobie naukę i oddzielił ją od swej wiedzy. Technikę stworzył zaś po to, aby ułatwić sobie pracę, która daje mu zaspokojenie potrzeb jego egzystencji: „I stworzył człowiek komputer i uznał, że jest dobry". Ważne jednak jest to, by dzięki jego wynalezieniu nie zabrnąć w ślepą uliczkę wariacji na temat tego urządzenia, by pozwoliło ono wyjść naszej cywilizacji na szerokie tory wiodące do lepszej przyszłości, co jednak nie oznacza, że namawiam PKP do zmiany szerokości naszych dróg żelaznych...

JERZY KLAWIŃSKI





obwiednia dźwięku (ang. *envelope*), czyli funkcja amplitudy sygnału dźwiękowego w czasie, określająca trzy podstawowe stadia dźwięku:

— stadium nabrzmiewania (ang. *attack*), w którym następuje przyrost natężenia dźwięku (jego tonów składowych) i ustalanie składu dźwięku.

— stadium ustalone (prawie ustalone), (ang. *sustain*), w którym skład dźwięku nie ulega zmianie.

— stadium wybrzmiewania (zanikania), (ang. *release*), czyli spadek natężenia dźwięku (tonów składowych) aż do jego zaniknięcia.

Stadia nabrzmiewania i wybrzmiewania nazywamy nie ustalonymi. Obwiednia jest podstawowym czynnikiem decydującym o rozpoznawaniu przez nas dźwięków instrumentów, bardziej nawet istotnym niż barwa dźwięku. Robiono doświadczenia z dźwiękiem instrumentów sztucznie pozbawionych obwiedni. Słuchacze mylili wówczas różne instrumenty ze sobą. Zresztą obwiednia kształtuje w dużym stopniu ogólne wrażenie barwy, w szczególności stadia nie ustalone. Nie wszystkie instrumenty posiadają stadium ustalone, różnią się też od siebie znacznie czasem nabrzmiewania. Najkrótszy czas nabrzmiewania mają instrumenty dęte blaszane oraz szarpane instrumenty strunowe (poniżej 20 ms), dłuższy — smyczkowe (40 do 120 ms), najdłuższy zaś posiadają np. flety (do 600 ms) i piszczałki organowe o niskim rejestrze (do 1 s). Czas wybrzmiewania jest zwykle dłuższy od narastania. Najkrócej wybrzmiewają instrumenty dęte (100—200 ms), dłużej smyczkowe (2—4 s) czy też fortepian (8—20 s).

W prostych syntezatorach elektronicznych stosuje się obwiednię czteroczęściową, przy czym omówione stadium nabrzmiewania dzieli się tu na dwa odrębne: narastania (ang. *attack*) i ustalania (ang. *decay*). W nowych instru-

MUZYKA SKOMPUTERYZOWANA NA COMMODORE C64

Jacek Jędrzejowski

Część 2

Artykuł ten, to dalszy ciąg rozpoczętej w poprzednim numerze „InforMika" próby opisu zagadnień z zakresu muzyki oraz zjawisk akustycznych związanych z syntezą dźwięku, a także możliwości wykorzystania dźwiękowych walorów komputera Commodore 64.

Dzisiaj zajmiemy się obwiednią, składem i widmem dźwięku.

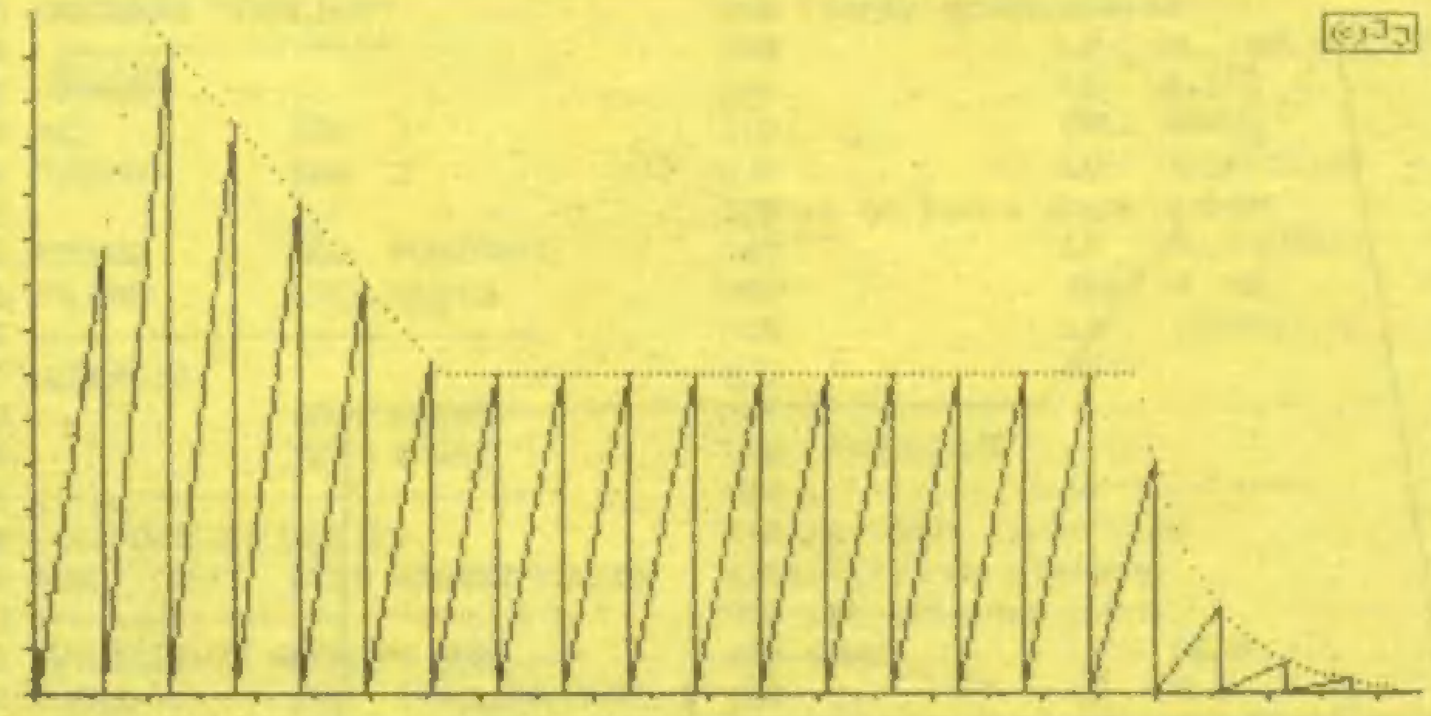
Obwiednia — syntezy dźwięku część pierwsza

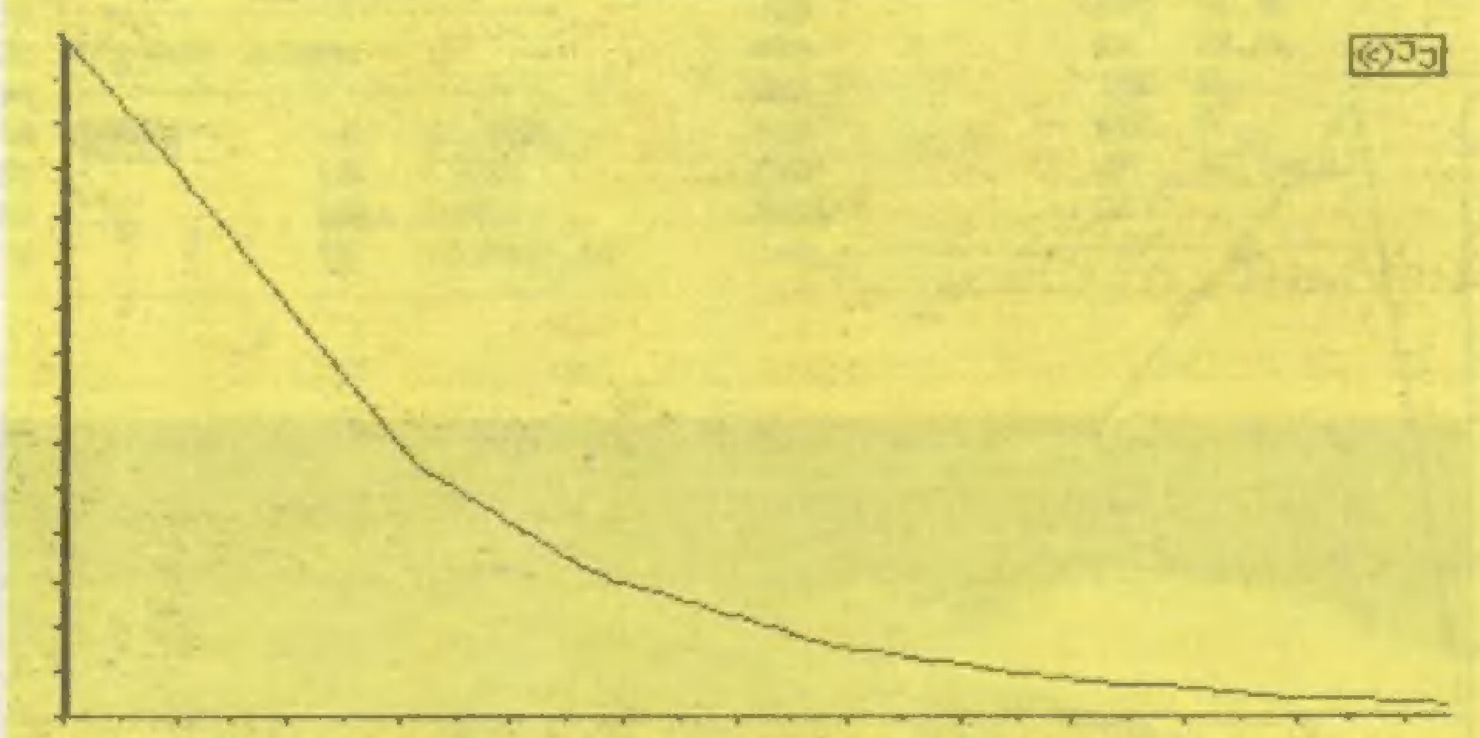
Przez syntezę dźwięku należy rozumieć zestawienie jego wielu parametrów fizycznych. Jednym z nich jest

mentach elektronicznych stosuje się 5 i 6 faz obwiedni a w dużych komputerowych systemach muzycznych można ją kształtować dowolnie. Obraz obwiedni naturalnych instrumentów stanowi bowiem dość skomplikowaną krzywą, zależną od wielu czynników, np. dobrego zestrojenia (zjawisko dudnienia) i wszelkie uproszczenia ograniczają swobodę syntezy dźwięku.

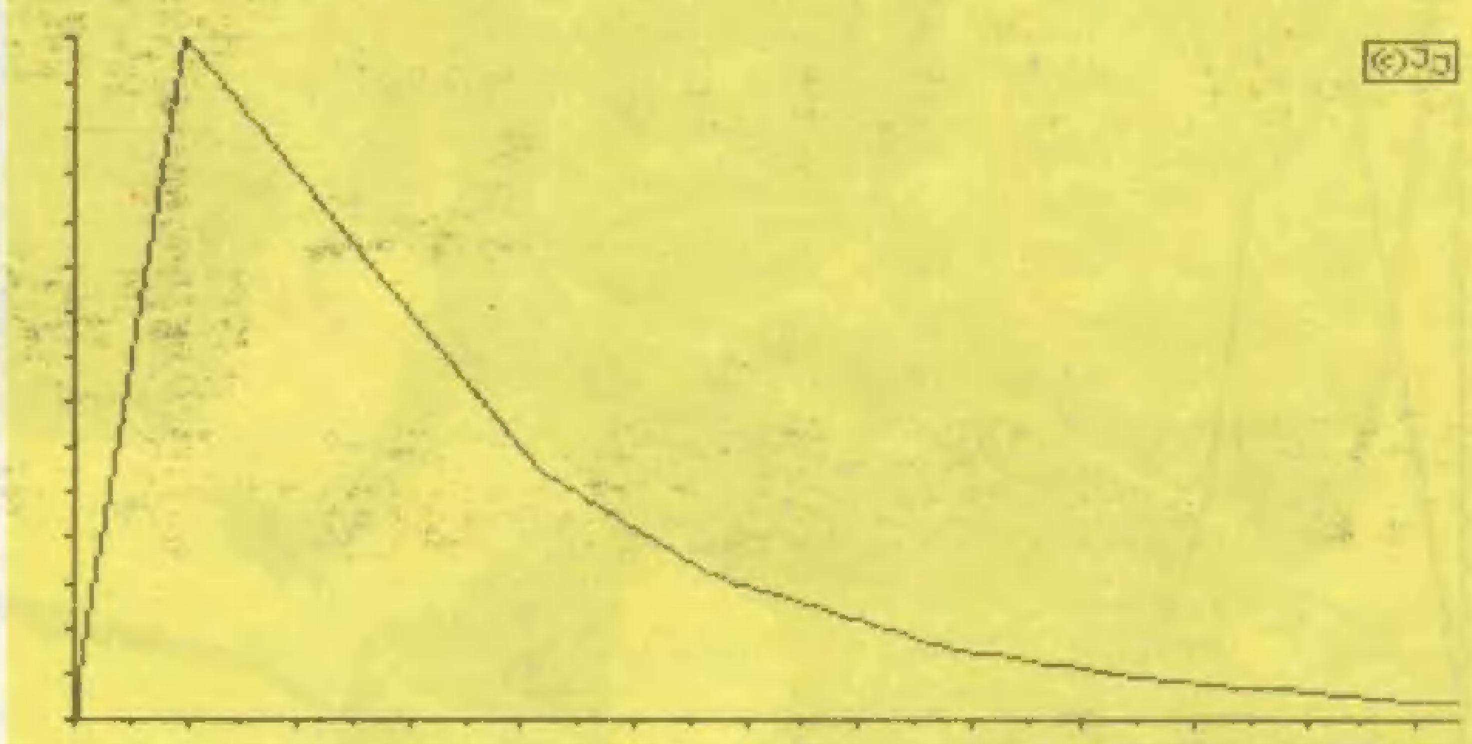
SID należy do prostych syntezatorów i ma cztery stadia obwiedni. Ich opis ilustrowany będzie rysunkami, które wykonane zostały na C64 przy pomocy prostego programu działającego na zasadzie oscyloskopu. W rejestrze 28 omawianego układu dźwiękowego możliwy

jest ciągły odczyt wartości liczbowej stanu generatora obwiedni dla kanału trzeciego. Są to wartości 8-bitowe (dziesiętnie 0—255). Posłużyły one do wykreślenia obwiedni w funkcji czasu. Oś X wykresów wyskalowana jest w dziesiątych częściach sekundy (grubsza kreska oznacza każde 100 ms). Oś Y podzielona jest na 16 poziomów amplitudy, co odpowiada szesnastu stopniom regulacji fazy *sustain*. Czas pomiaru obwiedni jest na wszystkich wykresach jednakowy i wynosi 1.25 sekundy. Pod wykresami znajdują się dane odpowiadające wartościom poszczególnych faz obwiedni wpisanym w rejestry syntezatora (posłużono się nazwami w jęz. angielskim jako najbardziej zwięzłymi i popularnymi od szeregu lat). W każdym z kanałów SID obok generatora sygnału znajduje się jeden generaor obwiedni oraz modulator amplitudy. Dane obwiedni doprowadzone są do modulatora amplitudy, który wg. nich reguluje amplitudę sygnału z generatora — w ten sposób słyszymy dźwięk o zmieniającym się w czasie natężeniu. Jest to ukazane na rys. A gdzie modulowany jest przebieg piłokształtny (wykres przebiegu zrobiony sztucznie). Zajmijmy się jednak samą obwiednią. Na rys. 1 obserwujemy dwa stadia nie ustalone — narastanie, które jest bardzo szybkie (jego wykres przylega do osi Y), oraz fazę wybrzmiewania. Nie jest to jednak faza *release* — faza *decay*. Tu staje się jasne, dlaczego podzielono nabrzmiewanie na dwie części. Otóż jeśli brak stadium ustalonego *sustain*, to stadium ustalania się dźwięku *decay* opada aż do zera, stając się w ten sposób fazą wybrzmiewania (słowo faza i stadium będą tu używane zamiennie). Faza *attack* rozpoczyna się w chwili ustawienia bitu bramkującego, omówionego poprzednio. Na rys. 2 dźwięk narasta w czasie 100 ms, następna faza pozostaje bez zmian, ulega tylko przesunięciu. Jeśli impuls bramkujący zakończy się przed dojściem ostatniego stadium do zera, to zostanie ono gwałtownie zakończone, jak to widać na **rysunku 3.** Jeśli zdecydujemy się na stadium ustalone (*sustain*), to będzie ono trwało tak długo, jak impuls bramkujący, po czym amplituda dźwięku opadnie gwałtownie aż do zera **(rys. 4)**. Faza ustalania się dźwięku (*decay*) uzupełnia tu we właściwy sposób fazę narastania (*attack*), tworząc w ten sposób stadium nabrzmiewania. Warto tu zwrócić uwagę, że jeśli impuls bramkujący jest odpowiednio długi, to faza narastania dochodzi zawsze do punktu szczytowego amplitudy generatora obwiedni. Tutaj zaczyna się faza ustalania (*decay*), która zdąża do poziomu fazy ustalonej (*sustain*). Jeśli poziom ten jest równy zeru, to ustalanie jest zarazem wybrzmiewaniem. Jeśli nie, to faza ustalania kończy się na poziomie stadium ustalonego (*sustain*), tzn. dalsza część tej fazy zostaje „obcięta" **(rys. 5)**. Stadium ustalone kończy się w momencie wyzerowania bitu bramkującego. Wówczas rozpoczyna się właściwe stadium wybrzmiewania (*release*) kończące obwiednię. Na **rys. 5** jest ono znikomo krótkie (6 ms), a na **rys. 6** wynosi ok. 300 ms. Tak właśnie wygląda pełna obwiednia fortepianu. Tutaj chciałbym posłużyć umuzykalnionym programistom i skomputeryzowanym muzykom radą: nie używajmy zbyt długich czasów trwania stadiów obwiedni dla krótkotrwałych z założenia dźwięków, bo obwiednia będzie realizowana w szczątkowej formie. Chodzi tu konkretnie o fazy narastania (*attack*) i ustalania (*decay*), bowiem faza ustalona trwać może dowolnie długo (czy też krótko), a wybrzmiewanie następuje dopiero po za-

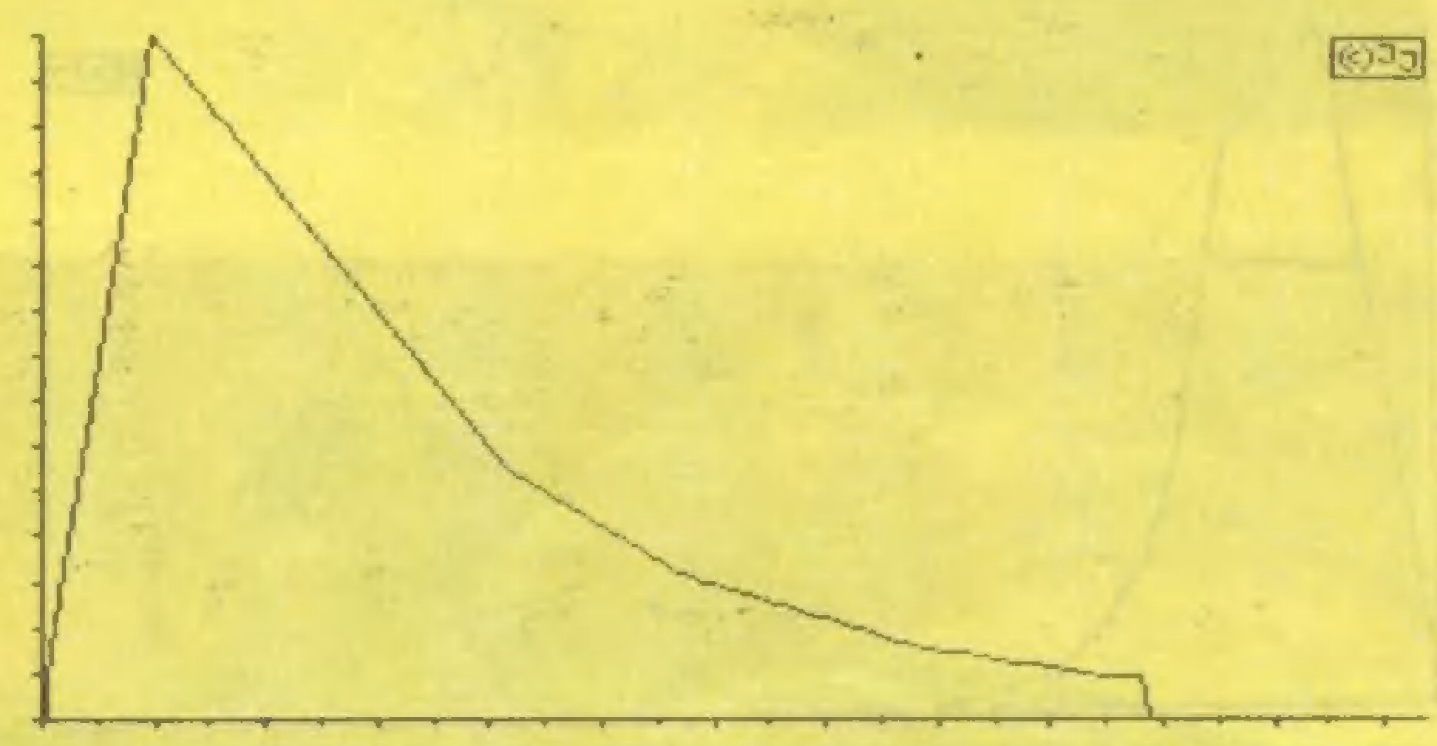




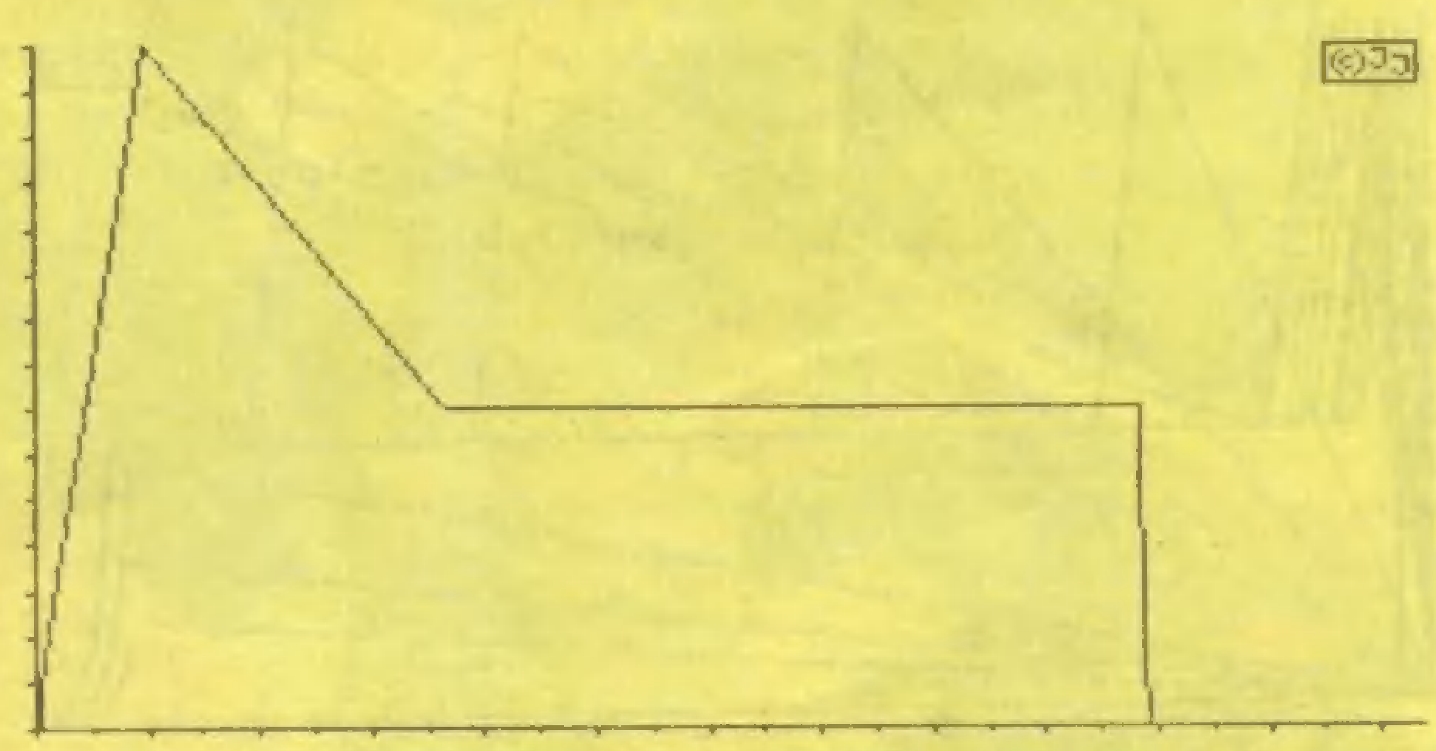
ATTACK = 0 ; DECAY = 10 ; SUSTAIN = 0 ; RELEASE = 0



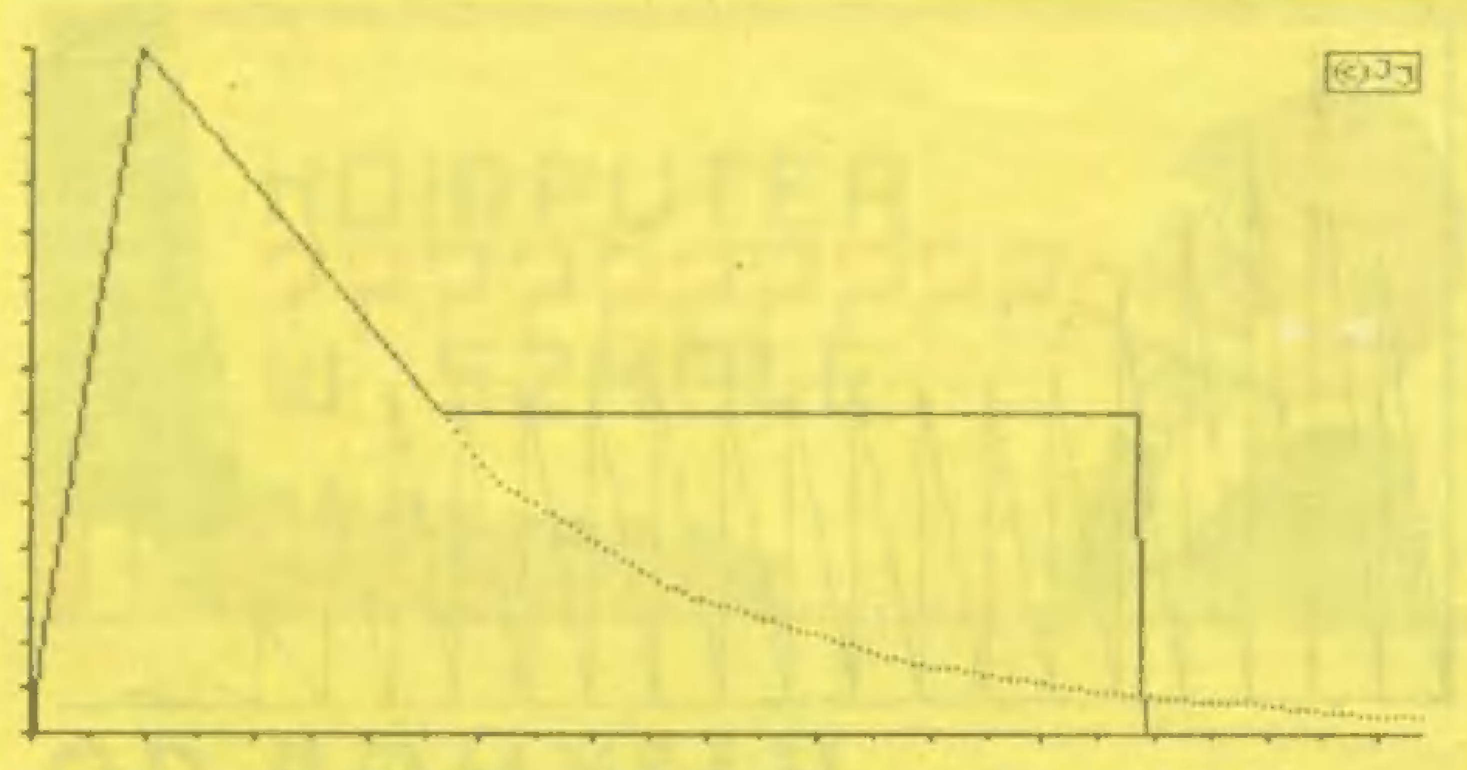
ATTACK = 8 ; DECAY = 10 ; SUSTAIN = 0 ; RELEASE = 0



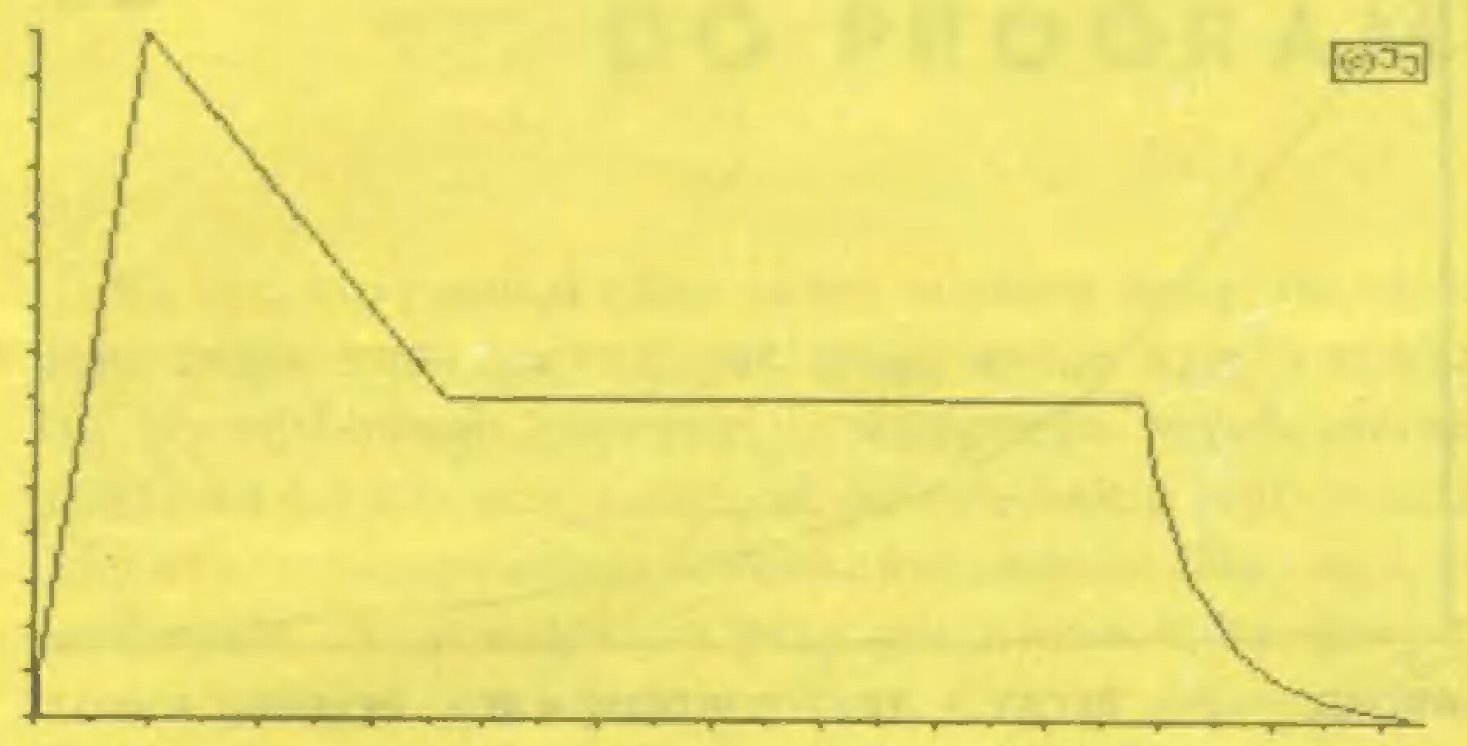
ATTACK = 8 ; DECAY = 10 ; SUSTAIN = 0 ; RELEASE = 0



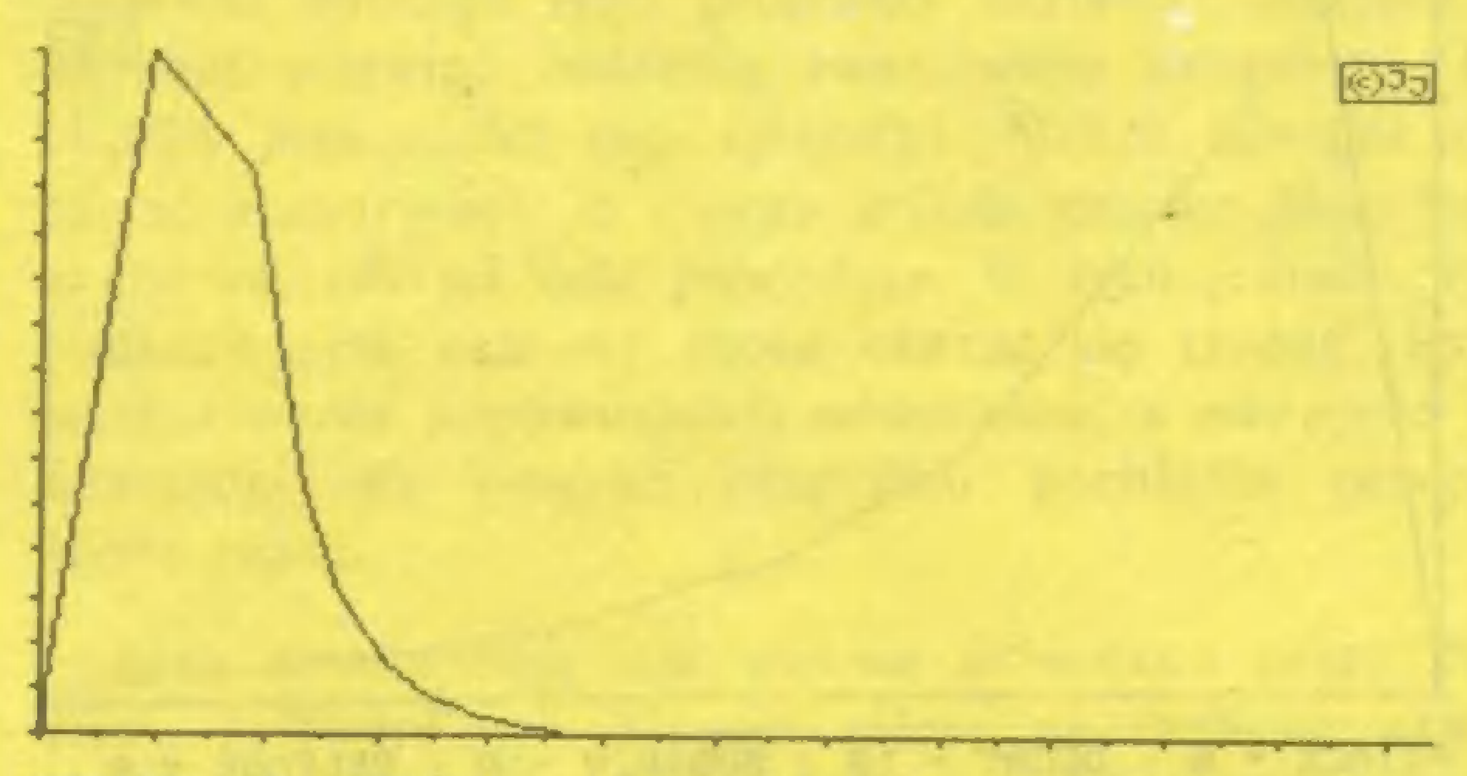
ATTACK = 8 ; DECAY = 10 ; SUSTAIN = 7 ; RELEASE = 0



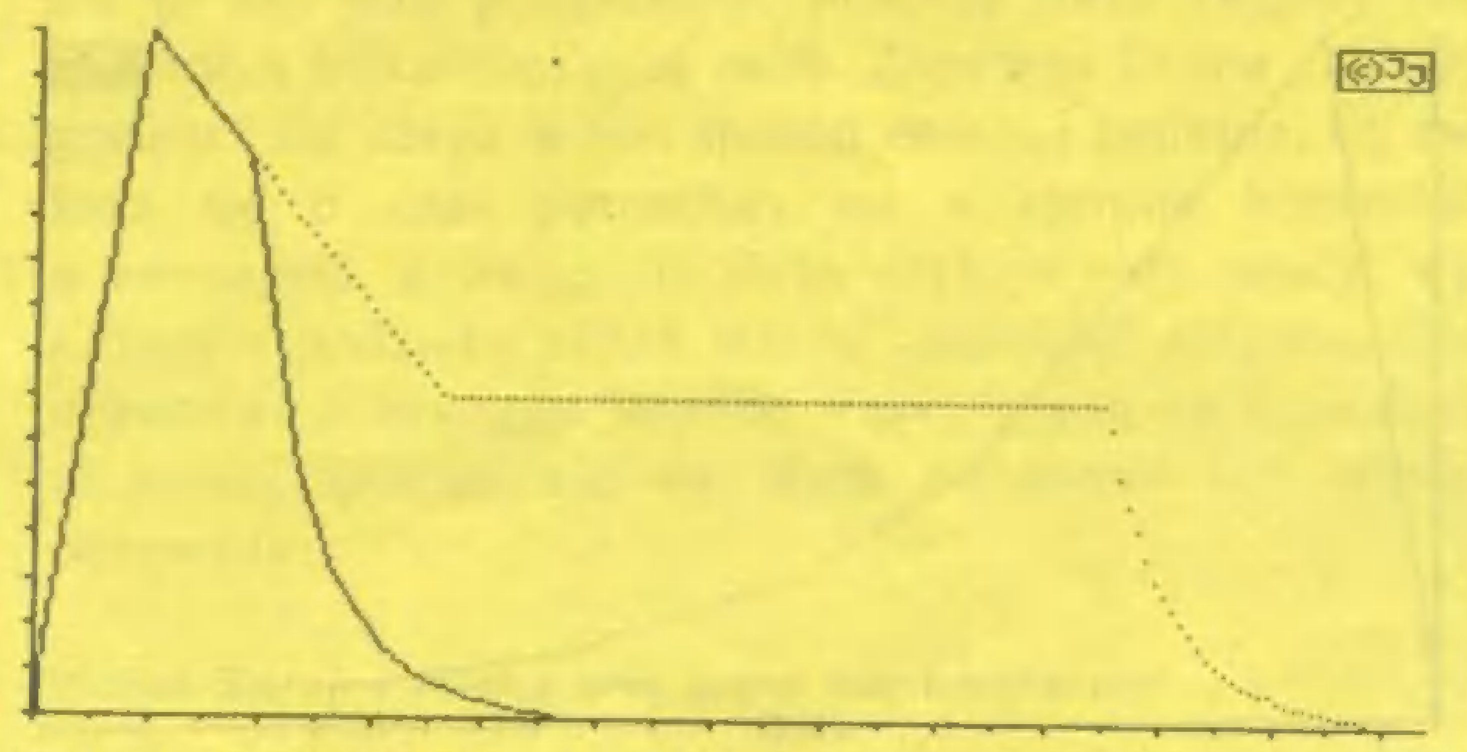
ATTACK = 8 ; DECAY = 10 ; SUSTAIN = 7 ; RELEASE = 0



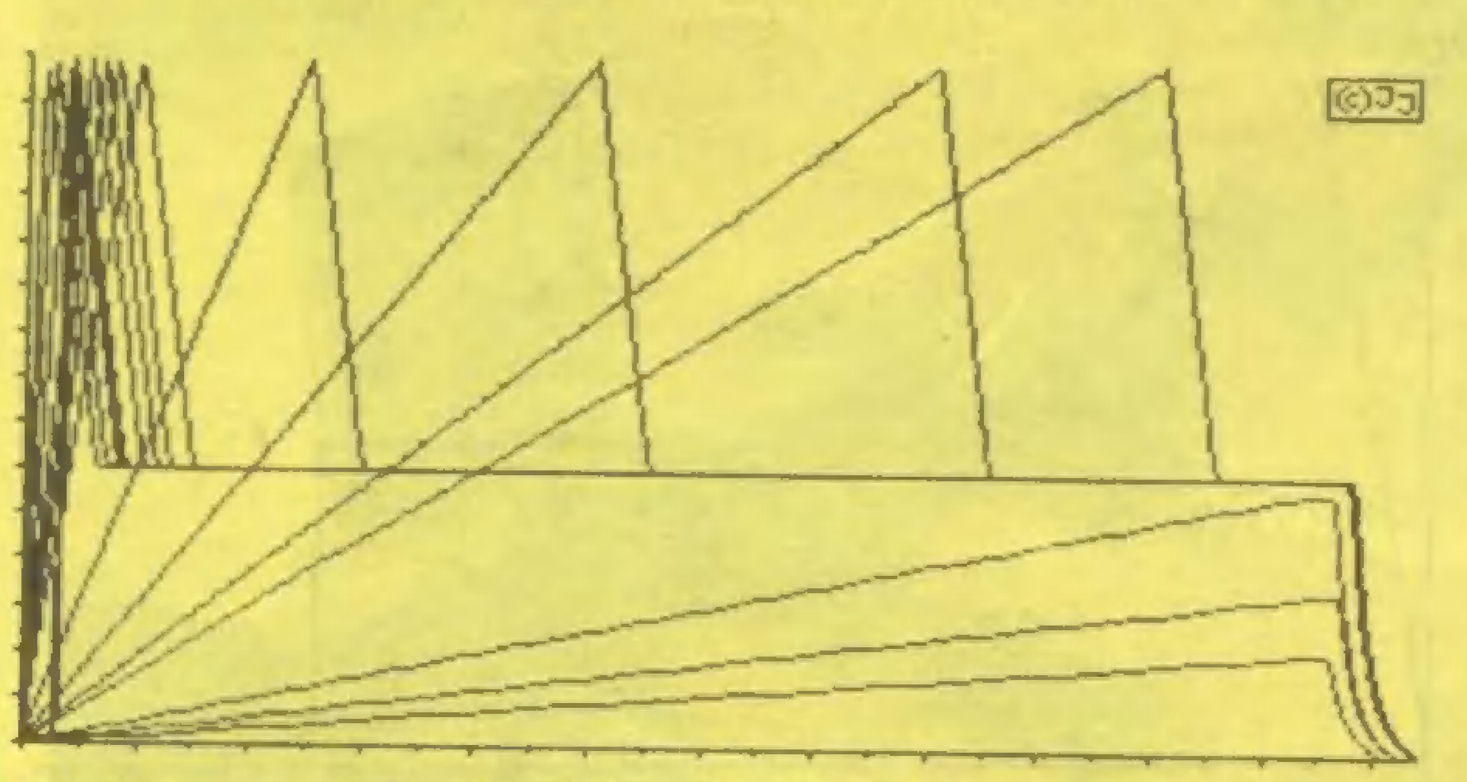
ATTACK = 8 ; DECAY = 10 ; SUSTAIN = 7 ; RELEASE = 8



ATTACK = 8 ; DECAY = 10 ; SUSTAIN = 7 ; RELEASE = 8



ATTACK = 8 ; DECAY = 10 ; SUSTAIN = 7 ; RELEASE = 8



ATTACK = 0 - 15 ; DECAY = 7 ; SUSTAIN = 6 ; RELEASE = 3

kończeniu impulsu bramkującego (po zwolnieniu klawisza w instrumencie). Taki przypadek obserwujemy na rys. 7, gdzie impuls bramkujący kończy się po upływie 200 ms i w tym momencie faza ustalania przechodzi w fazę wybrzmiewania (*release*). Po prostu stadium ustalania nie osiąga poziomu ustalonego (rys. 8).

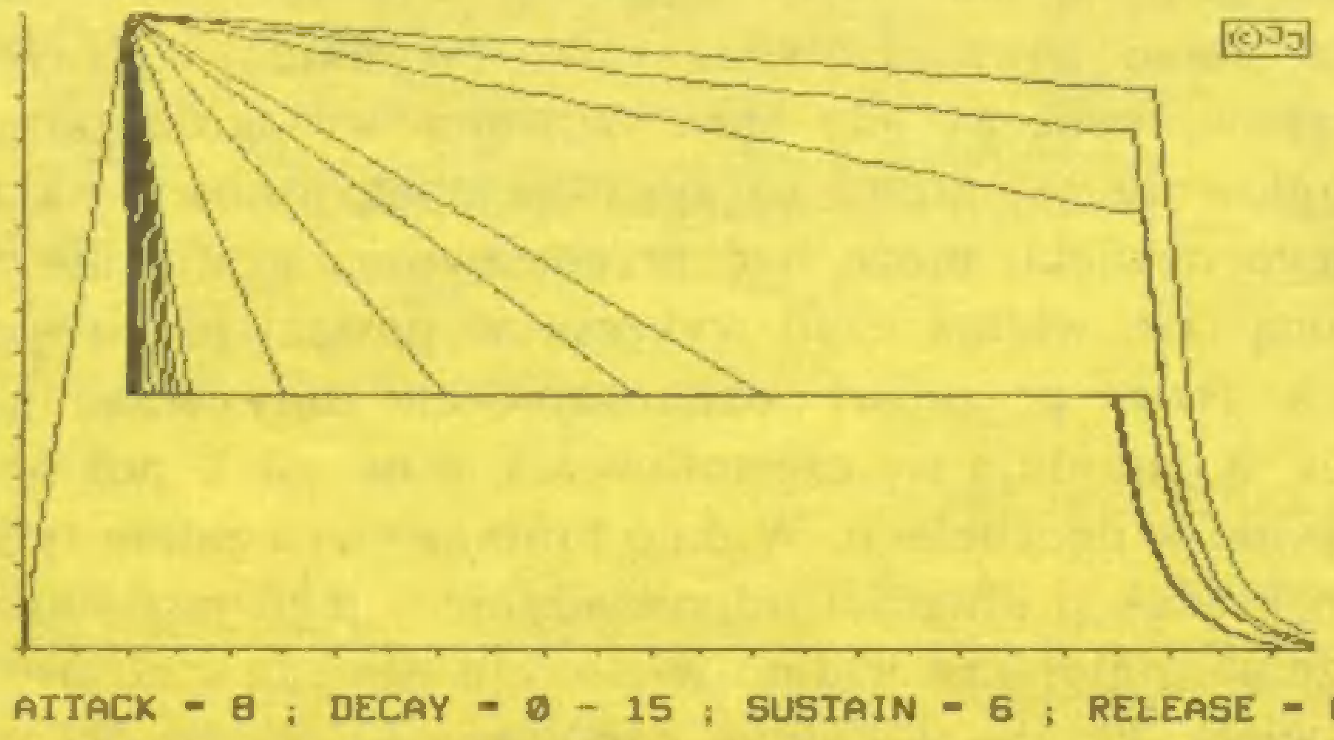
A oto jak regulujemy czasy stadiów obwiedni w układzie SID. W kanale 1 rejestr nr 5 (numeracja od zera!) służy do ustawiania faz *attack* i *decay* jednocześnie. Podobnie rejestr nr 6 decyduje o fazach *sustain* i *release*. Cztery starsze bity rejestru regulują pierwsze ze stadiów, a cztery młodsze — drugie. Każde ze stadiów posiada 16 stopni regulacji (od 0 do 15). W przypadku fazy ustalonej decydują one o **poziomie amplitudy** podtrzymania dźwięku, zaś w trzech pozostałych — o **czasie narastania**, czy też opadania dźwięku. Czasy dla faz *decay* i *release* są jednakowe. Wartości te podano w tabeli 1. Chcąc ustalić obwiednię mnożymy poziom regulacji ustawiany starszym półbajtem przez 16 i dodajemy do wymaganej wartości poziomu ustawianego młodszym półbajtem (np. *attack* = 8, *decay* = 10 : $8*16+10=128+10=138$). Postępujemy tak wówczas, gdy posługujemy się rozkazem POKE w języku BASIC (linia 150 programu). Do regulacji obwiedni w kanałach 2 i 3 służą odpowiednio rejestry 12, 13 oraz 19, 20.

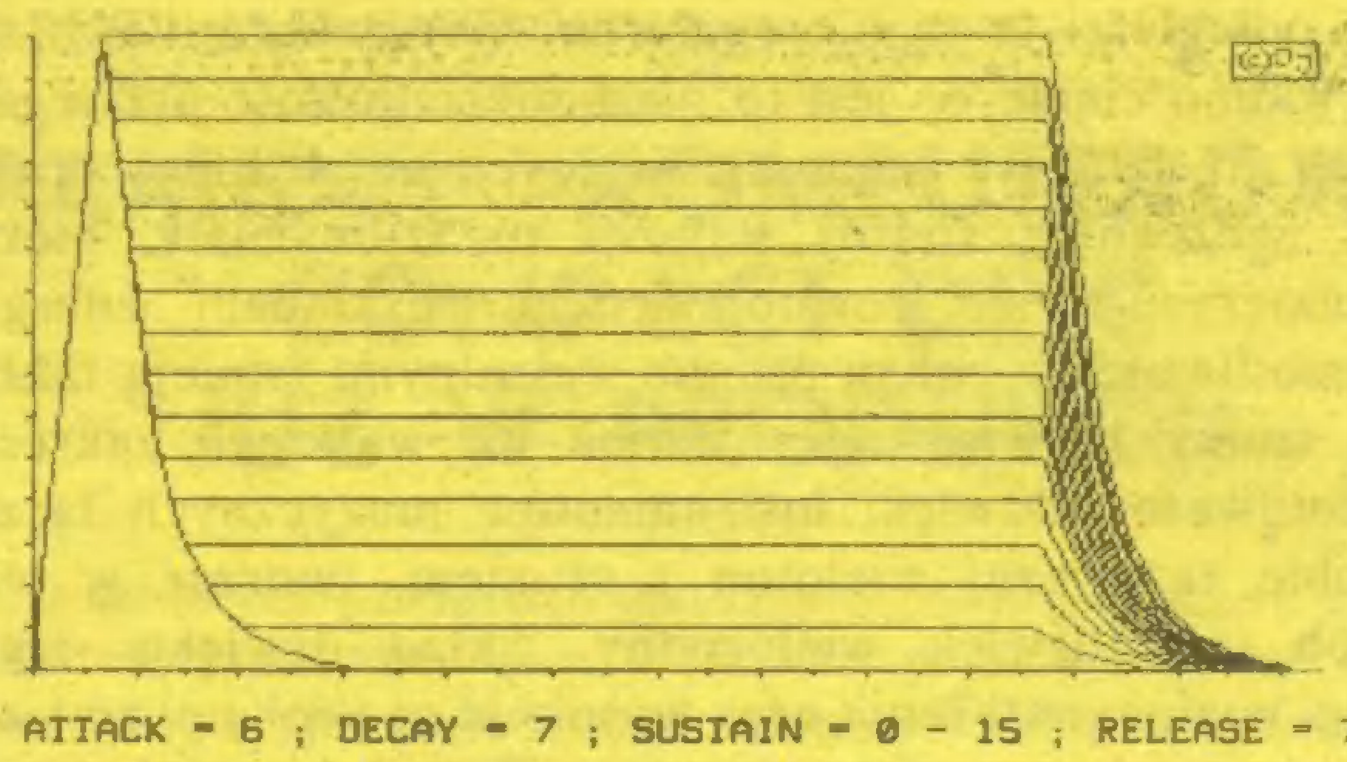
Na rysunkach **9**, **10**, **11**, i **12** uwidocznione są rodziny charakterystyk obwiedni układu SID dla kolejnych jej stadiów. W przypadku narastania i ustalania trzy ostatnie charakterystyki nie zmieściły się w całości na ekranie komputera (długie czasy). Podobnie jest ze stadiami wybrzmiewania. Zastosowanie dłuższego czasu pomiaru spowodowałoby utratę wyrazistości charakterystyk początkowych (o krótkich czasach trwania).

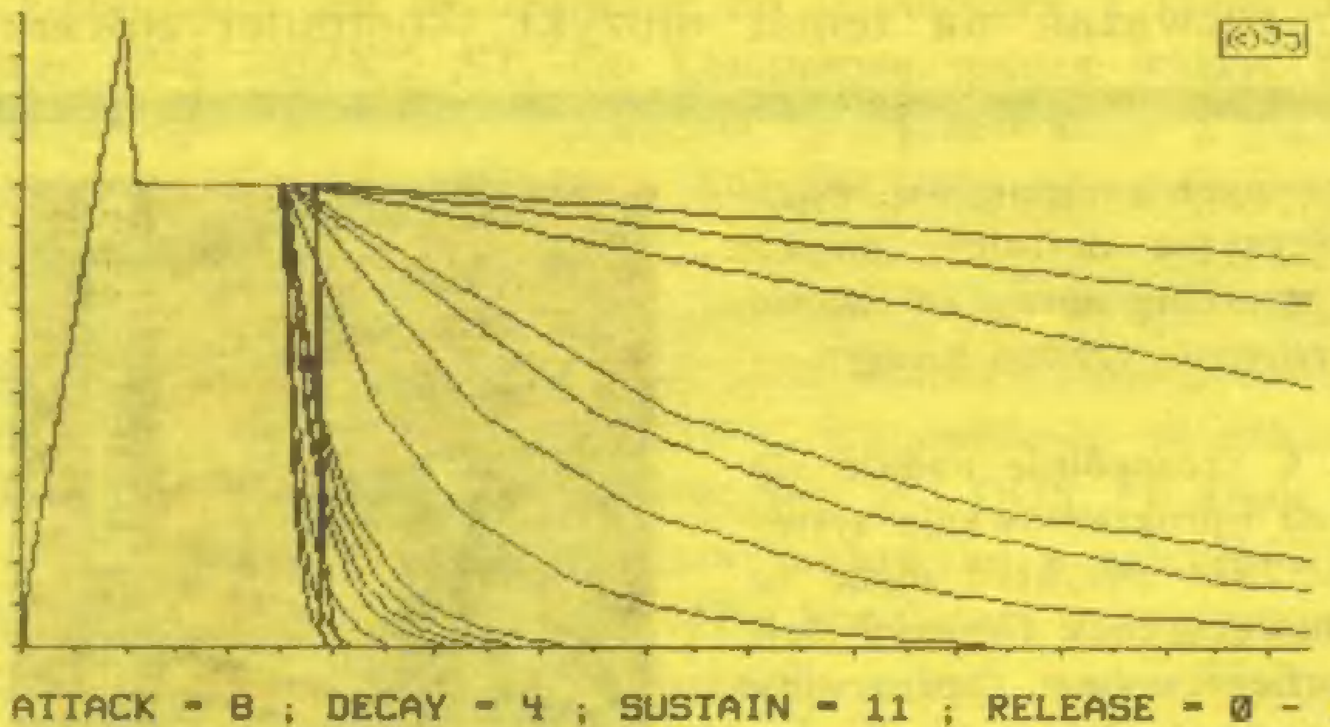
Na zakończenie tego rozdziału chciałbym dodać, że istnieje jeszcze **stadium przejścia** od jednego dźwięku do drugiego, w którym skład dźwięku ulega stopniowym przemianom od stanu ustalonego pierwszego dźwięku do stanu ustalonego drugiego dźwięku. W naszym syntezatorze ma to miejsce wtedy, gdy w trakcie stadium wybrzmiewania (*release*) pojawi się następny impuls bramkujący, dlatego nie należy zastępować fazy *release* fazą *decay*, gdyż ta istnieje podczas stanu wysokiego impulsu bramkującego. Jeśli rezygnujemy ze stadium ustalonego, to należy ustawiać *decay* i *release* na tę samą wartość.

Tabela 1

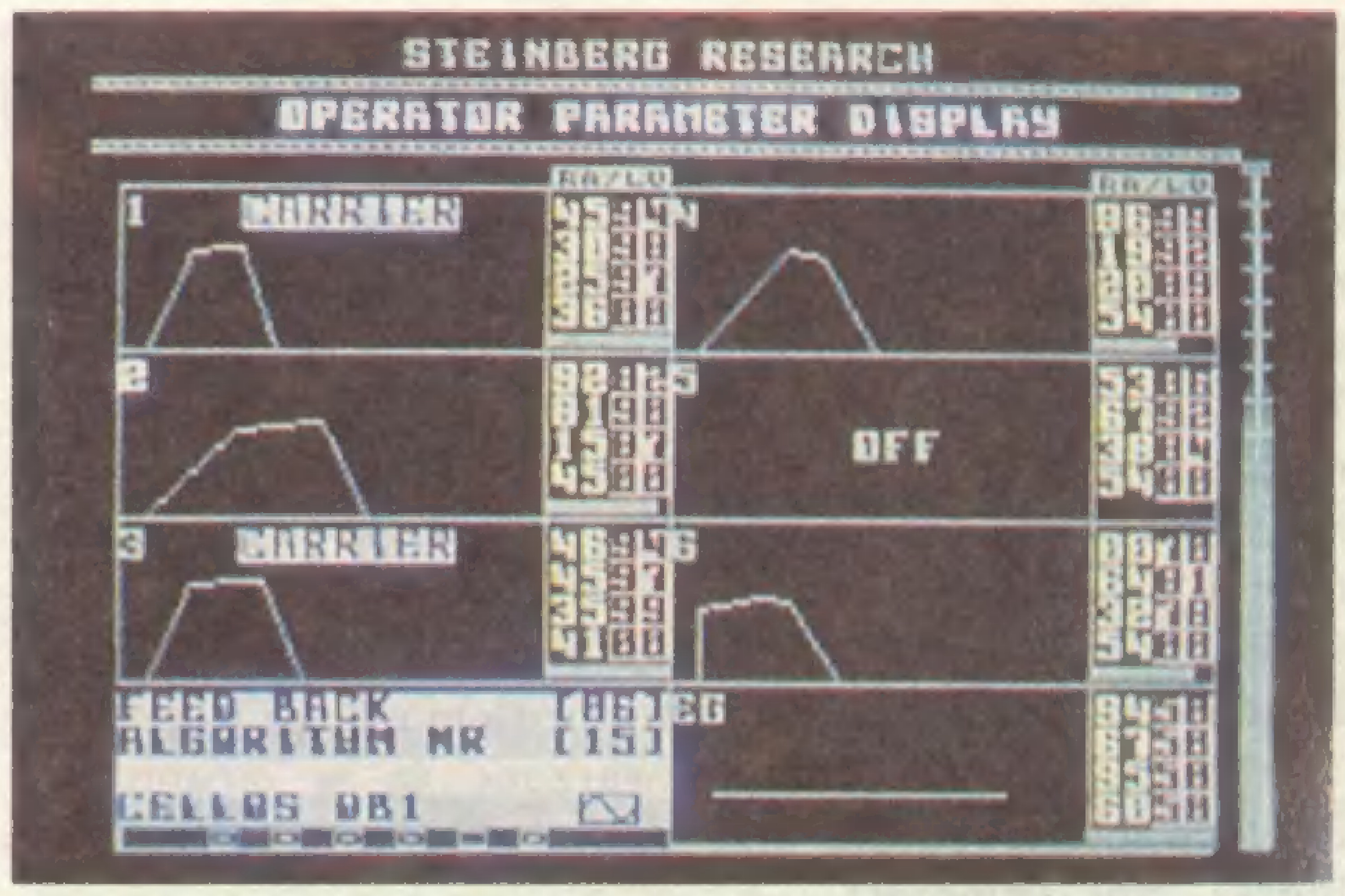
Wartosc		Narastanie (ATTACK)	Ustalanie i wybrzmiewanie (DECAY/RELEASE)
DEC	HEX	czas trwania fazy	czas trwania fazy
0	0	2 ms	6 ms
1	1	8 ms	24 ms
2	2	16 ms	48 ms
3	3	24 ms	72 ms
4	4	38 ms	114 ms
5	5	56 ms	168 ms
6	6	68 ms	204 ms
7	7	80 ms	240 ms
8	8	100 ms	300 ms
9	9	250 ms	750 ms
10	A	500 ms	1.5 s
11	B	800 ms	2.4 s
12	C	1 s	3 s
13	D	3 s	9 s
14	E	5 s	15 s
15	F	8 s	24 s

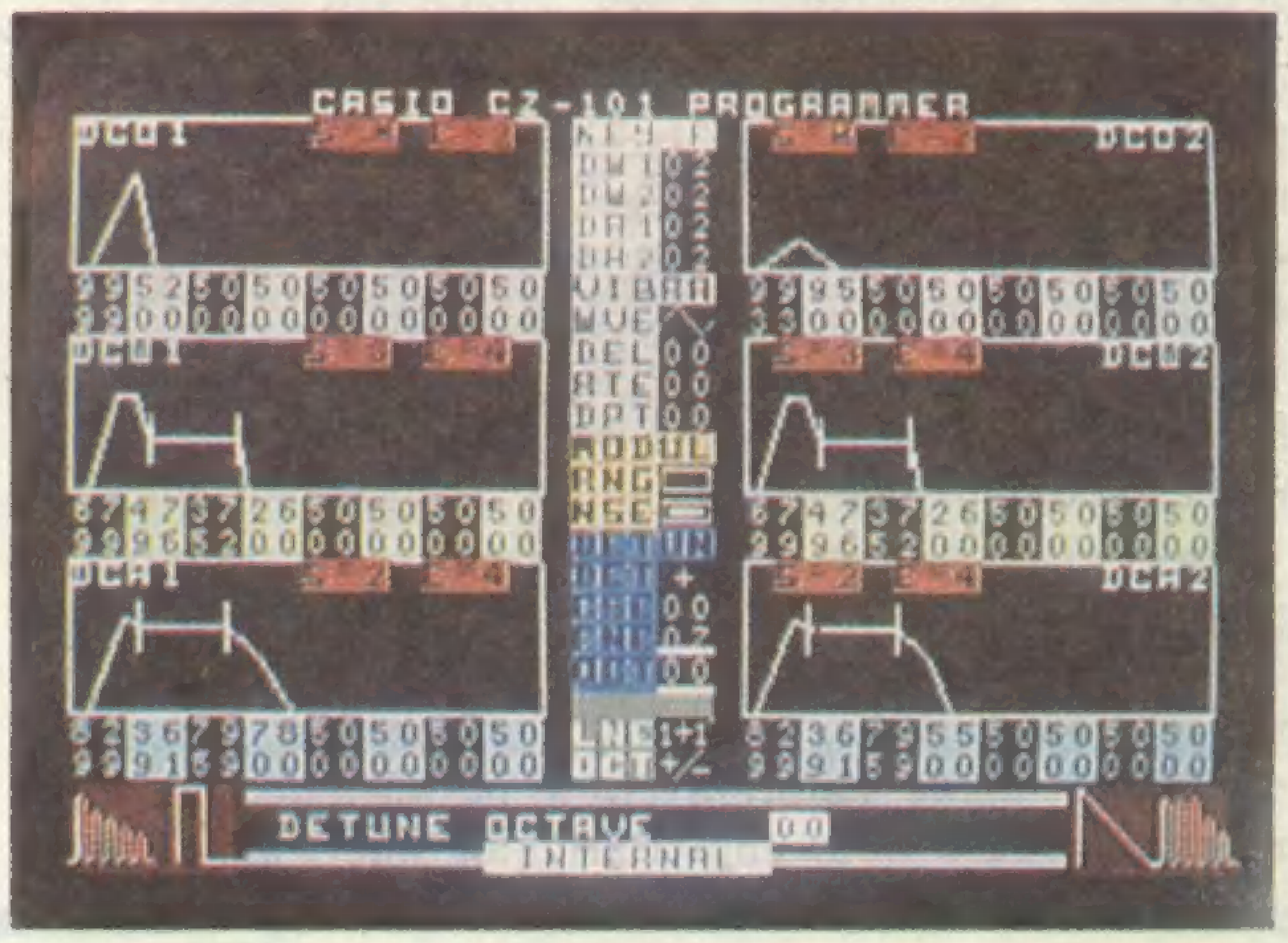


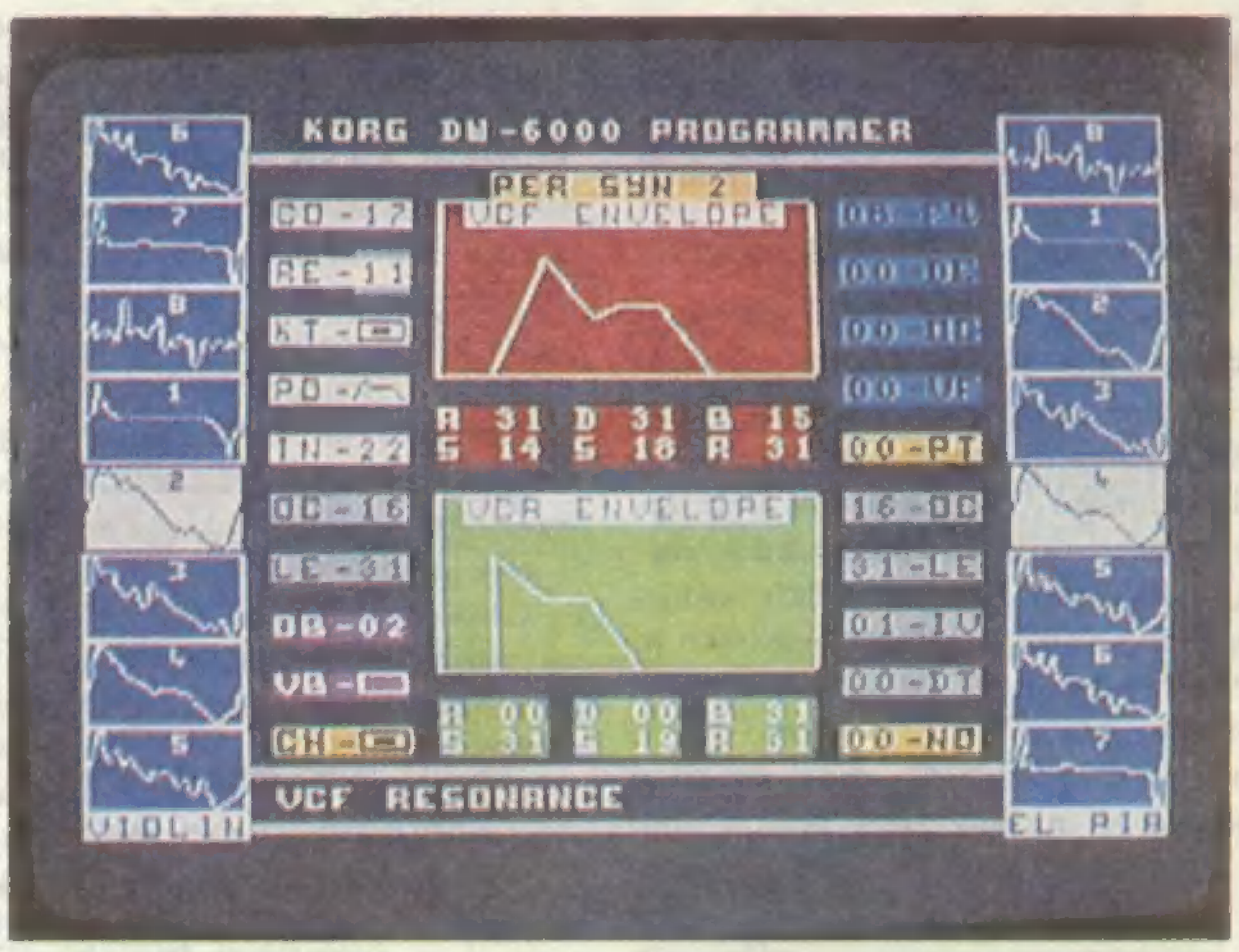




Po zakończeniu impulsu bramkującego krzywa *release* będzie kontynuacją krzywej *decay* i możliwe jest wtedy płynne przejście do następnego dźwięku. Jeśli wartość parametru dla *release* równa jest zero, a dla *decay* nie, to krzywa zmieni kierunek, gwałtownie opadając i nie będzie fazy przejścia. Trzeba tu wspomnieć o pewnym zjawisku stanowiącym osobny problem w akustyce. Chodzi mianowicie o tzw. **efekt włączenia** i efekt wyłączenia, czyli bezpośrednie przejście od stanu ciszy do dźwięku i odwrotnie. Pojawia się wówczas słyszalny trzask jako rezultat fragmentu drgania o przebiegu prostokątnym, które posiada nieskończoną ilość nieparzystych harmonicznych. Im bardziej strome jest zbocze tego drgania, tym silniejszy i dłużej trwający jest szum (trzask). Zjawisko to występuje, gdy zamiast przełączania tylko bitu bramkującego włączamy i wyłączamy na przemian generator (o czym już wspominałem) lub gdy niewłaściwie kończymy obwiednię. W przypadku SID najkrótsza faza wybrzmiewania możliwa do ustawienia ma czas trwania 6 ms (*release* = 0) — jest to wystarczająco długo, by trzask nie był słyszalny (przy wyzerowaniu bitu bramkującego).







Ekrany komputera Commodore 64 podczas pracy 3 edytorów służących do tworzenia programów muzycznych sterujących cyfrowymi syntezatorami dźwięku. Od góry: Sound Editing Software dla syntezatora Yamaha DX 7, COSMO CASIO CZ EDITOR dla syntezatora Casio i Bankloader and Editor dla syntezatora Korg DW 6000

Skład i widmo dźwięku

Dotychczas wielokrotnie pojawiało się pojęcie **składu dźwięku**. Dźwięk bowiem, w odróżnieniu od tonu (który praktycznie nie występuje w naturalnych warunkach), jest wypadkową wielu tonów składowych. Nazywamy go **wielotonem** jeśli każdy z jego tonów składowych da się wyodrębnić i jeśli pomiędzy tonami sąsiadującymi w skali częstotliwości są miejsca „puste . Jeżeli zaś częstotliwości składowych jest bardzo wiele i wypełniają one gęsto skalę

częstotliwości, wówczas dźwięk ten nazywamy **szumem**. Spośród wielotonów pewien ich rodzaj jest dla nas obecnie najważniejszy. Wyróżniają się one następującą cechą szczególną: częstotliwości tonów składowych tworzą szereg wielokrotności jednej częstotliwości podstawowej, zwany **szeregiem harmonicznym** lub **szeregiem alikwotów**. Wielotony tego rodzaju nazywamy **wielotonami harmonicznymi** lub **współtonami**, a ich tony składowe **tonami harmonicznymi** lub **alikwotami**. Tak spokrewnione tony tworzą przebieg będący ich wypadkową — co prawda nie sinusoidalny, ale okresowy. Częstotliwość najniższego tonu składowego jest zarazem częstotliwością całego wielotonu. Wielotony harmoniczne wytwarzane są w strunowych i dętych instrumentach muzycznych oraz w naturze (głos ludzki, głos zwierząt).

Na omówionym powyżej zjawisku fizycznym opiera się cała **harmonia muzyczna** czyli nauka o dźwiękach współbrzmiących i zasadach ich łączenia w akordy. W **szeregu harmonicznym** (od pierwszego do szesnastego alikwotu) występują wszystkie **interwały muzyczne**, a ich położenie w szeregu ma wpływ na pojęcie konsonansu i dysonansu (współbrzmienia „zgodnego i „niezgodnego"). Tonom harmonicznym (alikwotom) współtonu możemy nadać numery porządkowe w ten sposób, że każdemu alikwotowi odpowiada numer będący ilorazem jego częstotliwości przez częstotliwość najniższego tonu. Stąd związki między alikwotami wyrażają się w liczbach całkowitych (lub ich odwrotnościach). Interwały zdefiniowane w ten sposób noszą nazwę **naturalnych** w odróżnieniu od wspomnianych już w poprzednim artykule **interwałów temperowanych**. **Skład dźwięku** jest jedną z jego cech fizycznych określaną jako ilość, wysokość i poziom natężenia tonów składowych. Dotyczy ona wszystkich rodzajów dźwięków, podczas gdy sprecyzowaną wysokość posiadają głównie współtony (oczywiście obok tonów). Skład każdego dźwięku może być przedstawiony graficznie za pomocą tzw. **widma** czyli wykresu w postaci pionowych kresek (tzw. prążków) oznaczających tony składowe. Na osi X znajdują się częstotliwości, a na osi Y poziomy natężenia w decybelach. Widmo tonu zawiera zatem tylko jeden prążek o długości odpowiadającej jego poziomowi natężenia, natomiast widmo wielotonu posiada tych prążków wiele. Prążki w widmie współtonu są rozmieszczone w równych odległościach (przy liniowej skali częstotliwości), podczas gdy w widmie wielotonu nieharmonicznego odległości te są nieregularne. Obraz szumu tworzy tzw. **widmo ciągłe** — jest to nieskończona ilość przylegających do siebie (a zatem zlewających się ze sobą) prążków. Szczególny rodzaj stanowi tu **szum biały**, który charakteryzuje się równomiernym rozkładem energii i częstotliwości w całym pasmie słyszalnym. Istnieją także tzw. **szumy barwne**, ograniczone do wąskiego zakresu częstotliwości. Dźwięki instrumentów muzycznych łączą w sobie zazwyczaj wieloton z szumem, tworząc w ten sposób tzw. **dźwięk wielorodny**. Skład dźwięku, jego łączny poziom natężenia oraz wysokość są cechami zmiennymi w czasie, tworząc szersze pojęcie **barwy dźwięku**. Pozwala nam ona na odróżnianie różnego rodzaju dźwięków. I właśnie barwie poświęcona będzie trzecia i ostatnia część rozważań na temat muzyki skomputeryzowanej.

CIEKAWE KSIĄŻKI

Brian W. Kernighan, Dennis M. Ritchie, JĘZYK C, Wydawnictwa Naukowo-Techniczne, 1987 r. Książka ta to typowy przykład wydawniczego „meteorytu". Dziesięciotysięczny zaledwie nakład w ciągu kilku dni (gdzieniegdzie nawet godzin) znikł z półek księgarskich. Nie bez powodu, gdyż jak twierdzi liczne grono specjalistów język programowania C ma wszelkie szanse w najbliższych latach stać się nowym standardem, robiąc karierę nie mniejszą niż w poprzednich trzech dziesięcioleciach Algol, Fortran, BASIC czy Pascal. Jak piszą w przedmowie tłumacze Danuta i Marek Kruszewscy książka ta jest podstawowym dokumentem całej bibliografii dotyczącej języka C. I nic dziwnego, ponieważ współautor książki D.M. Ritchie jest właśnie twórcą definicji C stworzonej w amerykańskiej firmie Bell Laboratories w 1972 r.



Główną zasadą, według której skonstruowany jest język C jest to, aby spełniając rolę języka wysokiego poziomu, z wszystkimi wynikającymi z tego konsekwencjami (łatwość tworzenia algorytmów i czytelnych programów użytkowych) w możliwie największym stopniu zbliżony był do sposobu rozwiązywania danego problemu przez komputer.

Z pozoru sprzeczne wymagania ułożenia języka jednocześnie wysokiego i niskiego poziomu znakomicie udały się zrealizować twórcom C. Nie dysponując wieloma udogodnieniami innych języków,C, jest jednak łatwo przyswajalny dla użytkownika, a jego bliskość wewnętrznemu językowi komputera przynosi takie korzyści jak prostota tworzenia kompilatorów i duża szybkość wykonywania programów. Operując typowymi obiektami, jak znaki, liczby i adresy C może być dostosowywany do pracy na rozmaitych typach komputerów. Poza tym precyzyjna definicja języka ułatwia wymianę oprogramowania między różnymi typami maszyn.

się nowym standardem, robiąc karierę nie mniejszą niż w poprzed-

Język C szczególnie nadaje się do pisania oprogramowania systemowego. Jako przykład może tu służyć noszący znak firmowy Bell Laboratories system operacyjny UNIX, napisany w języku C.

Książka napisana jest bardzo przystępnie i przeznaczona dla szerokiego grona czytelników. Sprzyja temu przejrzysty układ treści ilustrowanej częstymi przykładami. Po wprowadzeniu podstawowych pojęć w rozdziale pierwszym zatytułowanym „Elementarz autorzy omawiają zagadnienia typów, operatorów, wyrażeń, sterowania, struktury programów, wskaźników, tablic i operacji wejścia/wyjścia. Ostatni rozdział dotyczy środowiska systemu UNIX. Bardzo cenny jest umieszczony na końcu książki obszerny dodatek będący syntetycznie opracowanym przewodnikiem języka C.

Książka została wydana w USA w 1978 r, a więc przed rewolucją mikrokomputerową. Teraz należałoby sobie życzyć pojawienia się na naszym rynku książki omawiającej wersje języka C dla popularnych mikrokomputerów, szczególnie dla IBM PC.

(jn)



Dragan Tanaskoski, Stevan Milinković, Vladimir Janković, COMMODORE ZASVA VREMENA, Mikro Knjiga, Belgrad 1986. Książkę tę polecamy użytkownikom popularnego C64, którzy będą mieli okazję kupić ją w Jugosławii. Wbrew pozorom lektura tej pozycji, a w zasadzie bieżące korzystanie z niej przy obsłudze mikrokomputera Commodore 64, mimo istniejącej bariery językowej jest dość łatwe. Język serbsko-chorwacki jest dość podobny do polskiego, a przy korzystaniu z niewielkiego słowniczka można sobie całkiem nieźle radzić.

Książka jest przydatna zarówno dla zaawansowanego jak i początkującego użytkownika. Dla tego

Dokończenie na str. 25

DRUGA MŁODOŚĆ KARTY GRAFICZNEJ HERCULES

Roland Wacławek

Karta graficzna Hercules dla komputerów klasy PC/XT/AT nie pochodzi z firmy IBM, a mimo to stała się uznanym standardem w zakresie grafiki monochromatycznej dla komputerów osobistych tej firmy. Dlaczego?

Standardowa karta grafiki barwnej CGA może wprawdzie współpracować z monitorem jednobarwnym, lecz oferuje za małą rozdzielczość w pionie (tylko 200 linii rastra), co sprawia, że liniowa struktura ekranu jest wyraźnie widoczna. Poza tym w trybie tekstowym na jeden wiersz tekstu wypada tylko 200/25 = 8 linii rastra, co stwarza konieczność zagęszczenia wierszy i pogarsza czytelność znaków, wkomponowanych w siatkę 8*8 punktów. Standardowa karta monochromatyczna MGA firmy IBM daje świetną czytelność tekstu (każdy znak jest zbudowany w siatce 9*14 punktów), lecz nie dysponuje trybem graficznym.

Karta Hercules może emulować MGA w trybie tekstowym, ale oprócz tego oferuje tryb graficzny o rozdzielczości 720*348 punktów. W trybie tym podział obrazu na linie zanika, zaś jego czytelność jest wyśmienita. Poza tym karta Hercules dysponuje dwoma równorzędnymi blokami pamięci obrazu (po 31320 bajtów każdy), które w razie potrzeby można przełączać. Pozwala to np. rysować nowy rysunek, nieprzerwanie wyświetlając na ekranie jego starą wersję. Warto dodać, że Hercules ma wbudowany interfejs równoległy Centronics, co nieraz pozwala zrezygnować z zakupu dodatkowej karty interfejsów.

Karta Hercules, podobnie jak MGA, wymaga specjalnego, monochromatycznego monitora o podwyższonej częstotliwości odchylania poziomego, wyposażonego w tzw. wejście TTL. Sygnały: synchronizacji poziomej, synchronizacji pionowej, obrazu i poziomu jaskrawości są przesyłane oddzielnymi przewodami w postaci cyfrowej. Rozwiązanie to gwarantuje wielką stabilność i równocześnie doskonały kontrast obrazu, ale utrudnia samodzielne zbudowanie lub adaptację monitora w oparciu o standardowy sprzęt telewizyjny.

Karta Hercules została zbudowana wyłącznie w oparciu o standardowe układy scalone. Funkcję sternika obrazu pełni w niej popularny układ scalony Motorola 6845 — ten sam, co w karcie CGA. Skopiowanie karty Hercules nie stanowiło więc żadnego problemu. Dziś jest ona dostarczana w różnych mutacjach przez dziesiątki wytwórców, a jej cena jest porównywalna z ceną karty CGA. Wszystko to przyczyniło się do jej wielkiego rozpowszechnienia.

Jakie są mankamenty karty Hercules? Główna wada polega na tym, że... nie wylansowała jej firma IBM! Wskutek tego grafika typu Hercules nie jest uwzględniona w standardowych trybach pracy monitora. Co za tym idzie, włączenie zwykłego trybu tekstowego typu MGA jest możliwe w drodze zwykłego wywołania procedur w pamięci BIOS-ROM, ale tryb graficzny trzeba uruchamiać za pomocą specjalnych programów. Co za tym idzie, procedury w pamięci BIOS „nie wiedzą", że ekran pracuje w trybie Hercules, i nie mogą tej informacji przekazać programom użytkowym. Co gorsza, rejestry karty Hercules, służące do przełączania trybów graficznych, pozwalają tylko na zapis. Nie można więc w prosty sposób ustalić, w jakim trybie karta pracuje w danej chwili! Do czego to prowadzi, można przeczytać w artykule o kopii ekranu w tym samym numerze.

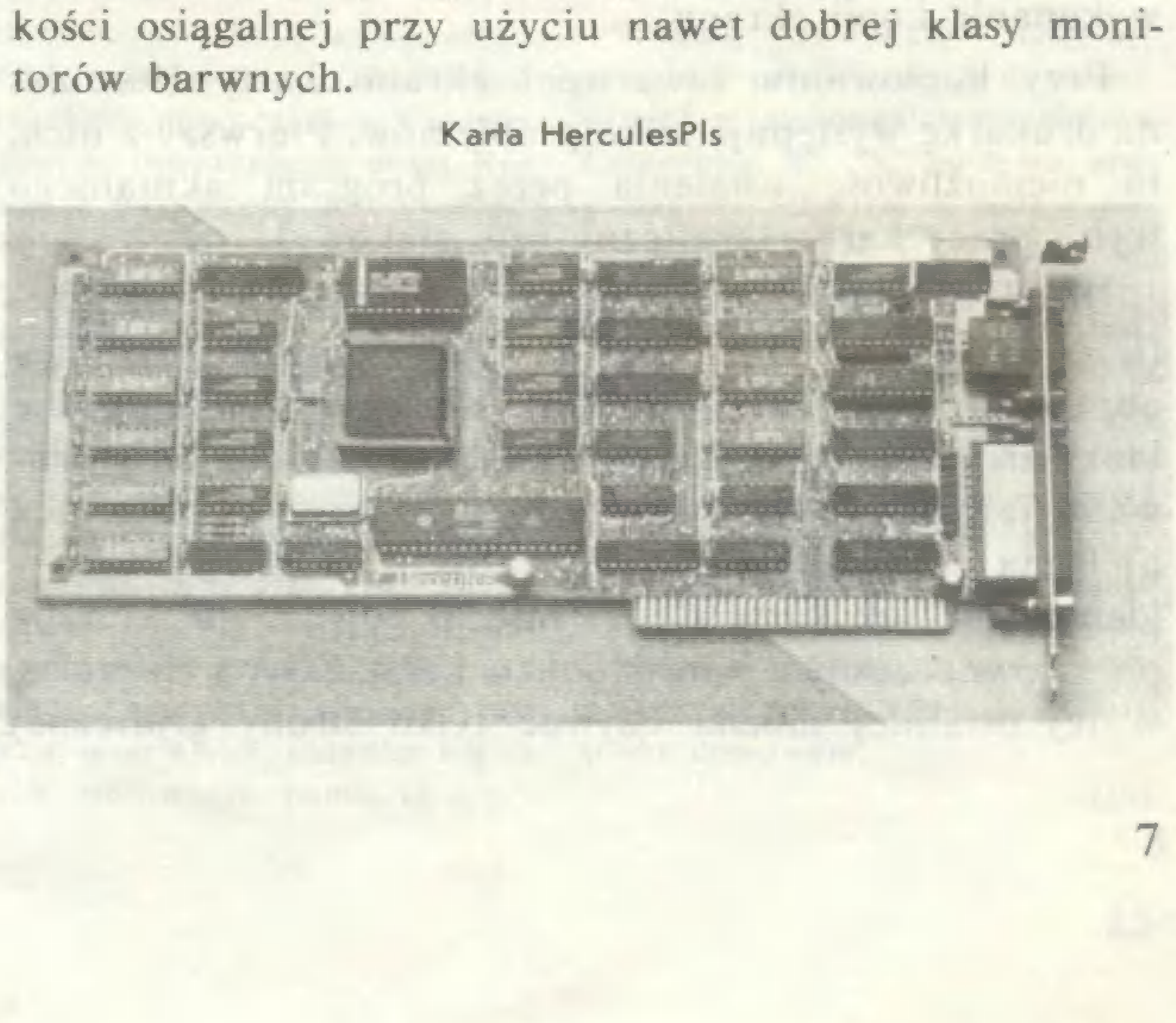
W ostatnim czasie na rynku pojawiła się nowa monochromatyczna karta graficzna Hercules Plus. W trybie graficznym ma ona podobne właściwości, jak stara karta Hercules. Istotna — szczególnie w naszych, polskich warunkach — jest natomiast możliwość definiowania własnych znaków w trybie tekstowym.

Stara karta Hercules miała wzorce znaków zapisane na stałe w pamięci ROM o pojemności 8 KB. Próba zmiany niektórych z nich, np. w celu uwzględnienia polskich znaków diakrytycznych (ą, ę, ł, itd.) wymagała wymiany tej pamięci i pociągała za sobą nieuchronną ingerencję we wnętrze komputera (znów ta lutownica!). Tymczasem Hercules Plus przechowuje wzorce znaków w pamięci RAM — można je tam zmieniać także w trakcie pracy programu. Karta ta ma 48 KB pamięci RAM, z czego w trybie tekstowym, jako pamięć ekranu, wykorzystanych jest tylko 4000 bajtów. Pozostałe 42 KB mogą przechowywać 12 różnych zestawów wzorców (wzorzec każdego znaku zajmuje 14 bajtów). Normalnie można korzystać tylko z jednego z nich. Przełączenie zestawu znaków powoduje natychmiastową zmianę wyglądu wszystkich liter na ekranie.

Wraz z kartą są dostarczane na dyskietce programy narzędziowe do projektowania własnych znaków, ich automatycznego ładowania do pamięci karty itd. Jeśli jeszcze stoi do dyspozycji drukarka z możliwością definiowania własnych znaków lub odpowiedni program sterujący, drukujący dowolne znaki w trybie graficznym, to kwestię znaków definiowanych przez użytkownika można uznać za ostatecznie rozwiązaną.

Czy karta Hercules Plus ma szanse w obliczu wzrastającego rozpowszechnienia karty EGA? Niewątpliwie tak. Po pierwsze, Hercules Plus jest rozwiązaniem znacznie tańszym (trzeba też uwzględnić monitor!). Po drugie, jakość tekstu na ekranie standardowego monitora monochromatycznego jest jednak (jak na razie) lepsza od jakości osiągalnej przy użyciu nawet dobrej klasy monitorów barwnych.

Karta HerculesPls



KOPIA EKRANU KARTY HERGULES NA DRUKARCE GRAFICZNEJ

Roland Wacławek

Do kopiowania zawartości ekranu na drukarkę służy w IBM-PC XT/AT, pracującym pod kontrolą systemu MS-DOS/PC-DOS, standardowe zlecenie klawiszowe [Shift] + [PrtSc]. W trybie znakowym funkcjonuje ono zarówno z kartą CGA, jak i MGA lub Hercules. Aby jednak móc uzyskać na ekranie kopię ekranu w trybie graficznym, potrzebny jest dodatkowy program, jak np. standardowy GRAPHICS.COM, dostarczany wraz z systemem operacyjnym. Niestety, GRAPHICS.COM pozwala uzyskać kopię ekranu graficznego tylko dla karty CGA. Użytkownicy karty Hercules są zdani na łaskę lub niełaskę programów użytkowych, które niekiedy dysponują własnymi procedurami kopiowania ekranu graficznego.

Niniejszy artykuł opisuje uniwersalny program HERCGRAK do kopiowania ekranu karty Hercules na drukarkę zgodną ze standardem IBM-GP (np. GLP-2, STAR GEMINI-10/15X1, SG-10/15, itd.). Aby dostosować go do współpracy z innymi drukarkami, wystarczy na ogół zmienić tylko kody sterujące trybem graficznym i wyborem odstępu międzyliniowego. Dla osób mających elementarne pojęcie o programowaniu w języku asemblera nie powinno przedstawiać trudności.

Program ten wystarczy wywołać tylko raz, najlepiej zaraz po załadowaniu systemu operacyjnego (najlepiej jego wywołanie wpisać do pliku AUTOEXEC. BAT). Ponieważ program musi znajdować się na stałe w pamięci operacyjnej i powinien zajmować jak najmniej miejsca, zaś wykonywane przezeń czynności są dość czasochłonne, został on zrealizowany w języku asemblera procesora 8088/86.

Powyższy program może współpracować praktycznie z każdym oprogramowaniem wykorzystującym kartę Hercules. Autorowi nie wiadomo o żadnym znanym programie, „nie znoszącym" się z programem HERCGRAK. Wyjątkiem są programy, które z zasady uniemożliwiają wykonanie kopii ekranu.

Przy kopiowaniu zawartości ekranu karty Hercules na drukarkę występuje kilka problemów. Pierwszy z nich, to niemożliwość ustalenia przez program aktualnego trybu pracy karty (graficzny czy znakowy).

Drugi to niemożliwość ustalenia, która ze stron graficznych jest w danej chwili wykorzystywana (pamięć obrazu karty Hercules jest podzielona na dwa identyczne bloki-stronice, lecz w danej chwili na ekranie odwzorowana jest zawartość tylko jednej ze stronic). Stronica nr 0 ma offset 0B000H, stronica nr 1 offset 0B800H — identyczny, jak karta CGA. Jest to powód, dla którego przy równoczesnym korzystaniu z karty CGA i Hercules, w tej ostatniej można używać tylko strony graficznej

```
;Kopia  ekranu graficznego karty HERCULES  na drukarce
;zgodnej z IBMGP; Roland Wacławek, Siemianowice Śl. 86
Kod_prog     SEGMENT
             ASSUME  CS:Kod_prog, DS:Kod_prog

MS_DOS       EQU  21H       ;nr przerw.  wywołującego DOS
PrtSc_Wekt   EQU  5         ;nr przerw. kopiowania ekranu
Monitor      EQU  10H       ;nr przerw. obsługi  monitora
Podaj_Stat   EQU  0FH       ;kod odczytu statusu monitora
Emisja_TTY   EQU  0EH       ;kod emisji znaku  (tryb TTY)
DrukarKa     EQU  17H       ;nr przerw.  obsługi drukarki
Klawiatura   EQU  16H       ;nr przer. obsługi klawiatury
TestujKlaw   EQU  1         ;nieniszczacy odczyt buf. Kl.
CzytajKlaw   EQU  0         ;normalny odczyt bufora klaw.
StatusKlaw   EQU  2         ;odczyt   statusu  klawiatury
CzytajWekt   EQU  35H       ;DOS - odczyt wektora przerw.
Ustaw_Wekt   EQU  25H       ;DOS - zapis  wektora przerw.
Zintegruj    EQU  27H       ;nr przerw. integracji progr.
Kod_CR       EQU  13        ;kod zn. powrotu kursora <CR>
Kod_LF       EQU  10        ;kod znaku zmiany linii  <LF>
Kod_BELL     EQU  7         ;kod ASCII, generujący dzwięk
Kod_ESC      EQU  27        ;kod symbolu Escape
Kod_F1       EQU  3B00H     ;dwubajtowy  kod klawisza  F1
Kod_F2       EQU  3C00H     ;dwubajtowy  kod klawisza  F2
Kod_F3       EQU  3D00H     ;dwubajtowy  kod klawisza  F3
Kod_Gr1920   EQU  5AH       ;kod trybu graficznego 1920p.
Kod_Gr960    EQU  4CH       ;kod trybu graficznego 960 p.
Kod_Ust_LF   EQU  33H       ;kod ustawiania odstępu linii
Graf_LF      EQU  24        ;odstęp wierszy  dla  grafiki
Tekst_LF     EQU  36        ;odstęp wierszy  dla  tekstów
Liczba_li    EQU  348       ;liczba linii ekranu HERCULES
Ost_Kolumn   EQU  719       ;maksymalna współrzędna osi X
Str_gr_0     EQU  0B000H    ;adres segmentu str. graf.  0
Str_gr_1     EQU  0B800H    ;adres segmentu str. graf.  1

             ORG   129
Parametry    LABEL BYTE     ;pole   parametrów  wywołania

             ORG   256
Inicjacja:   JMP   Zainstaluj

PrtSc_Adr:   DW    0          ;offset starej procedury PrtSc
PrtSc_Segm:  DW    0          ;segment starej proced.  PrtSc
PoczStanSP:  DW    0          ;początkowy stan  wskaźnika SP
Nr_StronyG:  DW    Str_gr_1   ;adres  aktualnej strony graf.
Skala:       DB    0          ;skala kopii:  0= 1:1,  1= 2:1
Flaga_Tryb:  DB    0          ;wskaźnik testowania trybu gr.
Flaga_Czek:  DB    0          ;wskaźnik oczekiwania  na  kl.
FlagaSHIFT:  DB    0          ;stan kl. [SHIFT] po wywołaniu
Granica:     DW    0          ;nr  ost. bitu różnego do zera

Herkules:    STI                           ;tutaj skok po rozkazie INT 5
             PUSH ES                       ;przechowaj  używane rejestry
             PUSH DS
             PUSH DX
             PUSH CX
             PUSH BX
             PUSH AX
             MOV  AX, Dane_DOS             ;umieść w  ES  adres segmentu
             MOV  ES, AX                   ;zmiennych systemowych MS-DOS
             CMP  BYTE PTR ES:PrtSc_Stat, 1
             JNZ  Nie_Kopia
Powrot_1:    JMP  Powrot                   ;kopiowanie ekranu  już  trwa

Nie_Kopia:   PUSH CS                       ;skopiuj do  DS zawartość  CS
             POP  DS
             MOV  AH, StatusKlaw
             INT  Klawiatura               ;ustal stan  klawiszy [SHIFT]
             MOV  BYTE PTR FlagaSHIFT, AL  ;i przechowaj ten stan
             TEST BYTE PTR Flaga_Tryb, 255 ;czy badać flagi BIOS?
             JZ   Grafika_ON               ;nie uwzgledniaj  flag   BIOS
             MOV  AH, Podaj_Stat           ;Kod funkcji BIOS dla prz. 10
             INT  Monitor                  ;Ustal  akt. status  monitora
             CMP  AL, 6                    ;uaktywniony  tryb graficzny?
             JZ   Grafika_ON               ;tak!  kopiowanie  graficzne
Copy_Tekst:  POP  AX                       ;przejdź do kopiowania tekstu
             POP  BX                       ;przechowaj  resztę rejestrów
             POP  CX
             POP  DX
             POP  DS
             POP  ES
             JMP  CS:DWORD PTR Prtsc_Adr   ;wywołaj  stary  PrtSc

Grafika_ON:  TEST BYTE PTR Flaga_Czek, 255;czy badać stan F1-F3?
             JZ   Nie_czekaj                ;nie - rozpocznij druk
             MOV  AH, Emisja_TTY
             MOV  AL, Kod_BELL
             INT  Monitor                  ;wyprowadź    krótki   dzwięk
Pob_Znak:    MOV  AH, CzytajKlaw           ;wywołaj obsługę klawiatury i
             INT  Klawiatura               ;czekaj na  wciśnięty klawisz

             CMP  AL, Kod_ESC              ;rej. AL :=odczytany kod ASCII
             JE   Powrot_1                 ;jeżeli ESC, zakończ  program
             MOV  DX, Str_gr_0             ;DX:=adres strony graficznej 0
             CMP  AX, Kod_F1               ;czy naciśnięto  klawisz F1 ?
             JE   Nr_podany                ;tak-uaktualnij nr str.  graf.
             MOV  DX, Str_gr_1             ;DX:=adres strony graficznej 1
             CMP  AX, Kod_F2               ;czy naciśnięto  klawisz F2 ?
             JE   Nr_podany                ;tak-uaktualnij nr str.  graf.
             CMP  AX, Kod_F3               ;czy naciśnięto  klawisz F3 ?
             JE   COPY_Tekst               ;tak - kopiuj w trybie tekst.
             JMP  Pob_Znak                 ;testuj kolejny znak zlecenia

Nr_podany:   MOV  WORD PTR Nr_StronyG, DX ;aktualizuj  str. graf.
Nie_czekaj:  MOV  WORD PTR PoczStanSP, SP;zapam. SP na wyp. błędu
Kopiuj:      MOV  AL, Graf_LF              ;AL :=odstęp linii dla grafiki
             CALL Ustaw_LF                 ;ustaw  odstęp między liniami
             CALL Ustal_StrG               ;ES:=adres segmentu akt.  str.
             MOV  AL, BYTE PTR FlagaSHIFT
             AND  AL, 00000001B            ;czy wciśnięty prawy [SHIFT]?
             JZ   Skala_1_1                ;prawy zwolniony:  skala   1:1
             JMP  Skala_2_1                ;prawy wciśnięty:  skala   2:1
```

nr 0. Jeśli chodzi o odpowiedniość między punktami ekranu, to każdej linii (720 punktów) odpowiada zwarty blok 90 bajtów. Rozmieszczenie poszczególnych bloków nie jest już jednak kolejne. Offset (przesunięcie) pierwszego bajtu, odpowiadającego linii o danej współrzędnej Y (0..347, licząc od góry), jest wyrażone wzorem: **8192*(Y MOD 4)+90*INT(Y/4)**, gdzie **MOD** symbolizuje operację modulo, zaś **INT** — część całkowitą (8192 = 2000H).

Trzeci problem polega na trudności w zapewnieniu odpowiedniej proporcji kopii, z uwagi na brak odpowiedniości między pionową i poziomą rozdzielczością ekranu oraz rozdzielczością standardowej drukarki graficznej. Drobne zniekształcenia są tutaj nieuniknione, lecz na ogół nie mają one istotnego znaczenia.

HERCGRAK może pracować w kilku trybach, wybieranych przełącznikami w chwili ładowania do pamięci. Oto dozwolone przełączniki i ich znaczenie:

0 Drukowana będzie kopia stronicy graficznej nr 0.
1 Drukowana będzie kopia stronicy graficznej nr 1.
/T Przed wydrukowaniem kopii program HERKOPIA sprawdzi wskaźnik trybu pracy ekranu w MS-DOS i w przypadku stwierdzenia trybu tekstowego kopia zostanie wykonana w zwykły sposób (opcja ta ma sens przy użytkowaniu programów informujących MS-DOS o trybie pracy ekranu).
C Po zażądaniu kopii ekranu naciśnięciem **[Shift]** + + **[PrtSc]** program HERKOPIA wyemituje krótki ton i poczeka na wciśnięcie jednego z czterech klawiszy: **[F1]**, **[F2]**, **[F3]** lub **[Esc]**. Oto funkcje tych klawiszy:

[Esc] : przerywa prace programu bez wykonania kopii
[F1] : drukuje kopie stronicy graficznej nr 0
[F2] : drukuje kopie stronicy graficznej nr 1
[F3] : drukuje kopie ekranu w trybie tekstowym.

Oto przykład wywołania programu: **HERCGRAK C**. Po każdym wywołaniu program „zapiszczy" i poczeka na wskazówki o aktualnym trybie pracy karty Hercules.

Aby spowodować skopiowanie ekranu, należy równocześnie nacisnąć klawisze **[Shift]** i **[PrtSc]**. Jeśli naciśnięty został lewy klawisz **[Shift]**, ekran będzie skopiowany w skali 1:2 (w poprzek arkusza). W razie naciśnięcia prawego klawisza **[Shift]** ekran zostanie odwzorowany w skali 1:1, czyli kopia będzie dwukrotnie większa, niż poprzednio (wydruk wzdłuż arkusza, pod kątem 90 stopni). Wydruk można w każdej chwili przerwać, naciskając klawisz **[Esc]**. Jakakolwiek niesprawność drukarki (brak papieru, niewłączenie do sieci itd.) powoduje zaniechanie wydruku bez dodatkowych komunikatów i innych konsekwencji.

Po uruchomieniu (etykieta: **Inicjacja**) program najpierw sprawdza, czy po nazwie programu nie wystąpiły jakieś przełączniki, a jeśli tak, ustawia odpowiednie wskaźniki wewnętrzne. Następna czynność (etykieta: **Kontyn 1**) to odczyt dotychczasowego wektora przerwania nr 5. Wydruk kopii ekranu jest bowiem uruchamiany właśnie rozkazem INT 5. HERCGRAK przechowa stary wektor na wypadek konieczności wykonania kopii ekranu w trybie tekstowym, zaś jako wektor przerwania INT 5 wpisze adres własnej procedury obsługi (etykieta: Herku-

```
Skala_1_1: XOR  DX,DX               ;początkowy numer wiersza = 0
Testuj_L:  MOV  CX,Ost_Kolumn-7     ;CX wskazuje ost. bajt linii
           PUSH DX                  ;zapamiętaj akt. numer linii
Testuj_B:  CALL Laduj_XY            ;ładuj do AL bajt pam. ekranu
           OR   AL,AL               ;czy zawiera ustawione bity?
           JNZ  Wypelniony          ;jeśli tak, trzeba drukowac !
           INC  DX                  ;ustaw nr nastepnego wiersza
           TEST DL,00000111B        ;osiagnieto granice bajtu?
           JNZ  Testuj_B            ;jeśli nie, testuj nast. bajt
           SUB  DX,08               ;odtwórz wyjściowy stan DX
           SUB  CX,08               ;przesun sie w lewo o 1 bajt
           JNS  Testuj_B            ;powtórz, jeśli jeszcze ekran
Drukuj_L:  CALL CR_LF               ;zmiana linii na drukarce
           POP  DX                  ;odtwórz wyjściowy stan DX
           ADD  DX,8                ;ustaw nastepna grupe linii
           CMP  DX,Liczba_li+1      ;czy linia już poza ekranem?
           JB   Testuj_L            ;jeśli nie, badaj nast. linie
           JMP  SHORT Koniec_K      ;kopiowanie ekranu zakonczone

Wypelniony:POP  DX                  ;wykryto wypełniony bajt
           PUSH DX                  ;przejdź do wydruku wiersza
           ADD  CX,8                ;poprawka na granice bajtu
           MOV  WORD PTR Granica,CX;zapam. liczbe bitów w lin.
           MOV  AX,CX               ;przenies liczbe bitów do AX
           CALL Inic_960            ;inicjuj grafike 960 p. w li.
           XOR  CX,CX               ;zeruj zawartosc rejestru CX
           CMP  DX,Liczba_li-8      ;czy osiagnieto ost. linie ?
           JA   Ostatnia_L          ;tak-wymaga spec. traktowania
Drukuj_B:  XOR  BX,BX               ;zeruj zawartosc rejestru BX
           PUSH DX                  ;pamiętaj DX jako współrz. Y
Dosun_Bit: CALL Point_XY            ;testuj punkt o X,Y w CX,DX
           ROL  BL,1                ;przesun bajt w BL o 1 w lewo
           OR   BL,AL               ;i dołacz doń kolejny bit
           INC  DX                  ;zwiększ o 1 współrzedna Y
           INC  BH                  ;odlicz kolejny dołaczony bit
           CMP  BH,8                ;czy opracowano już 8 bitów?
           JNZ  Dosun_Bit           ;jeśli nie,dołacz kolejny bit
           POP  DX                  ;odtwórz w DX pierwot. wsp. Y
           MOV  AL,BL               ;skopiuj kompletny bajt do AL
           CALL Drukuj              ;i wyślij go do drukarki
           INC  CX                  ;zwiększ o 1 współrzedną X
           CMP  CX,WORD PTR Granica  ;osiagnieto koniec linii?
           JNZ  Drukuj_B            ;nie - drukuj nastepny rządek
           JMP  SHORT Drukuj_L

Ostatnia_L:XOR  BX,BX               ;zeruj zawartość rejestru BX
           PUSH DX                  ;pamiętaj DX jako współrz. Y
Dosun_BitO:CALL Point_XY            ;testuj punkt o X,Y w CX,DX
           ROL  BL,1                ;przesuń bajt w BL o 1 w lewo
           OR   BL,AL               ;i dołacz doń kolejny bit
           INC  DX                  ;zwiększ o 1 współrzedną Y
           INC  BH                  ;odlicz kolejny dołączony bit
           CMP  BH,4                ;czy opracowano już 4 bity?
           JNZ  Dosun_BitO          ;jeśli nie,dołacz kolejny bit
           POP  DX                  ;odtwórz w DX pierwot. wsp. Y
           MOV  AL,BL               ;skopiuj kompletny bajt do AL
           SHL  AL,1                ;przesuń AL o 4 bity w lewo
           SHL  AL,1
           SHL  AL,1
           SHL  AL,1
           CALL Drukuj              ;wyślij bajt z AL do drukarki
           INC  CX                  ;zwiększ o 1 współrzedną X
           CMP  CX,WORD PTR Granica  ;osiagnieto koniec linii?
           JNZ  Ostatnia_L          ;nie - drukuj kolejny rządek
           JMP  SHORT Drukuj_L

Skala_2_1: XOR  CX,CX               ;początkowy numer kolumny = 0
Test_Linii:XOR  DX,DX               ;początkowy numer wiersza = 0


Test_Bajtu:CALL Laduj_XY            ;ładuj do AL bajt pam. ekranu
           OR   AL,AL               ;czy zawiera ustawione bity?
           JNZ  Nie_Pusty           ;tak! pora przejsc do wydruku
           INC  DX                  ;ustaw numer kolejnej kolumny
           CMP  DX,Liczba_li+1      ;badaj, czy jeszcze w ekranie
           JNZ  Test_Bajtu          ;jeśli tak, testuj nast. bajt
Nast_Linia:CALL CR_LF               ;koniec linii: wyślij CR,LF
           ADD  CX,8                ;powiększ o 8 numer kolumny
           CMP  CX,Ost_Kolumn+1     ;badaj, czy jeszcze w ekranie
           JNZ  Test_Linii          ;jeśli tak, badaj nast. linie
Koniec_K:  XOR  DH,DH               ;DH=0-status poprawnego konca
           JMP  Koniec              ;zakoncz wydruk kopii ekranu

Nie_Pusty: MOV  AX,Liczba_li+1      ;wpisz do AX nr linii granic.
           SUB  AX,DX               ;AX :=liczba linii do wydruku
           MOV  BX,AX               ;mnoz zawartosc AX przez 5,bo
           ADD  AX,AX               ;każdy bajt drukowany 5 razy
           ADD  AX,AX
           ADD  AX,BX
           CALL Inic_1920           ;inicjuj blok bajtów w tr.gr.
           MOV  WORD PTR Granica,DX;pam. nr pierw. niep. linii
           MOV  DX,Liczba_li        ;DX=nr ostatniej linii ekranu
Nast_Bajt: CALL Laduj_XY
           CALL Drukuj              ;powtórz 5 razy pod rząd
           CALL Drukuj              ;wydruk tego samego bajtu
           CALL Drukuj
           CALL Drukuj
           CALL Drukuj
           CMP  DX,WORD PTR Granica  ;osiagnieto koniec linii?
           JZ   Nast_Linia          ;tak - przejdz do nast. linii
           DEC  DX                  ;zmniejsz o 1 współrzedną Y
           JMP  Nast_Bajt           ;wyprowadz kolejny bajt

Inic_1920: PUSH CX                  ;włącza tryb 1920 p. w linii
           PUSH AX                  ;wyślij na stos liczbe bajtów
           MOV  CX,Kod_Gr1920       ;CL := kod inicjacji grafiki
           JMP  SHORT UstGrafik

Inic_960:  PUSH CX                  ;włącza tryb 960 pkt. w linii
           PUSH AX                  ;wyślij na stos liczbe bajtów
           MOV  CL,Kod_Gr960        ;CL := kod inicjacji grafiki
UstGrafik: MOV  AL,Kod_ESC
           CALL Drukuj              ;wyślij bajt z AL do drukarki
           MOV  AL,CL               ;kod inicjacji grafiki do AL
           CALL Drukuj              ;wyślij bajt z AL do drukarki
           POP  CX                  ;odtwórz w CX liczbe bajtów
           MOV  AL,CL               ;AL := młodszy bajt liczby
           CALL Drukuj              ;wyślij bajt z AL do drukarki
           MOV  AL,CH               ;AL := starszy bajt liczby
           CALL Drukuj              ;wyślij bajt z AL do drukarki
           POP  CX
           RET
                                    ;testuje, czy wciśnieto [ESC]
Testuj_ESC:MOV  AH,TestujKlaw       ;kod nieniszczącego odczytu
           INT  Klawiatura          ;jeśli bufor klawiatury pusty
           JZ   Brak_ESC            ;to na pewno nie podano [ESC]
           MOV  AH,CzytajKlaw
           INT  Klawiatura          ;odczytaj znak z bufora klaw.
           CMP  AL,Kod_ESC          ;czy wciśnieto klawisz [ESC]?
           JNZ  Testuj_ESC          ;nie - badaj nastepne znaki
           MOV  DH,0                ;kod poprawnego zakonczenia
           JMP  SHORT Przerwij

Drukuj:    PUSH DX                  ;wysyła do drukarki bajt z AL
           PUSH AX
           MOV  AH,0
           XOR  DX,DX               ;zeruj zawartosc rejestru DX
           INT  Drukarka            ;wyw. obsługe drukarki w BIOS
           TEST AH,00100101B        ;czy drukarka była sprawna?
           JNZ  Blad_Druku          ;drukarka niesprawna - koniec
           POP  AX
           POP  DX
Brak_ESC:  RET
```

les). Rozkaz INT 27H (INT Zintegruj) kończy pracę instalacyjnej części programu, czyniąc równocześnie procedurę **Herkules** rezydującą na stałe w pamięci RAM (integrując ją niejako z systemem operacyjnym).

Każda próba skopiowania ekranu spowoduje wywołanie procedury **Herkules**. Najpierw sprawdza ona, czy przypadkiem kopiowanie ekranu już się nie rozpoczęło. Wystarczy w tym celu sprawdzić zawartość odpowiedniej komórki w systemowej pamięci RAM. Jeśli kopiowanie już trwa, następuje niezwłoczny powrót z przerwania, w przeciwnym razie program analizuje ustawione przy inicjacji wskaźniki trybu pracy oraz bada, który z klawiszy **[Shift]** został wciśnięty: lewy czy prawy. Na tej podstawie wybierany jest tryb kopiowania i skala powiększenia. Wykonanie kopii obróconej o 90 stopni jest proste — wystarczy wysyłać do drukarki kolejne bajty pamięci ekranu. W przypadku kopii w poprzek arkusza należy jednak analizować pamięć ekranu bit po bicie i składać z poszczególnych bitów kolejne bajty.

Przed rozpoczęciem wydruku kopii graficznej HERCGRAM zmienia odstępy między liniami wydruku tak, aby poszczególne linie ściśle sąsiadowały ze sobą. Po zakończeniu wydruku albo po jego przedwczesnym przerwaniu odtwarzany jest standardowy odstęp, dający 6 linii na cal. Testowanie stanu klawisza **[Esc]** odbywa się po wyprowadzeniu każdej linii wydruku (program: **Testuj ESC**).

Aby uniknąć zbędnego wysyłania do drukarki bajtów zerowych na końcu linii, przed rozpoczęciem wydruku program analizuje każdą linię od tyłu, ustala położenie najbardziej wysuniętego, zaczernionego punktu i zapisuje jego położenie w komórce: Granica. Podprogram: **Laduj-XY** dokonuje niezbędnych przeliczeń współrzędnych ekranowych pod adresem bajtu w pamięci operacyjnej i ładuje odpowiedni bajt do rejestru AL. Podprogram: **Point XY** testuje natomiast stan pojedynczego punktu o podanych współrzędnych, ustawiając odpowiednio najmłodszy bit rejestru AL.



```
Blad_Druku:MOV   DH, OFFH             ;kod  zakoncz.  wskutek błedu
Przerwij:  MOV   SP, WORD PTR PoczStanSP  ;odtworz pierw stan SP
Koniec:    MOV   AX, Dane_DOS         ;umiesc w  ES  adres segmentu
           MOV   ES, AX               ;zmiennych systemowych MS-DOS
           MOV   AX, Tekst_LF         ;ustaw tekstowy odstep  linii
           CALL   Ustaw_LF
           CLI                        ;zablokuj przerwania  maskow.
           MOV   ES:PrtSc_Stat, DH    ;zapisz status operacji PrtSc
Powrot:    POP   AX                   ;odtworz  pierwotny stan rej.
           POP   ■
           POP   CX
           POP   DX
           POP   DS
           POP   ES
           IRET                       ;powrot z obsługi przerwania


CR_LF:     PUSH AX                    ;drukuje sekwencje <CR>, <LF>
           MOV   AL, Kod_CR
           CALL  Drukuj               ;wyślij zaw.  AL do drukarki
           MOV   AL, KOD_LF
           CALL  Drukuj               ;wyślij zaw.  AL do drukarki
           CALL  Testuj_ESC           ;test, czy  próba przerwania
           POP   AX
           RET
                                      ;ustawia  odstep  miedzylin.
Ustaw_LF:  PUSH AX                    ;przechowaj  wartosc odstępu
           MOV   AL, Kod_ESC
           CALL  Drukuj               ;wyślij kod <ESC>
           MOV   AL, Kod_Ust_LF
           CALL  Drukuj               ;wyślij kod ustawiania odst.
           POP   AX
           CALL  Drukuj               ;wyślij nową wartosc odstępu
           RET
                                      ;ładuje do AL bajt  pam. ekr.
Laduj_XY:  MOV   BH, DL               ;opisujacy punkt  (X=CX, Y=DX)
           ROR   BH, 1                ;mnoz  zawartosc  BH  przez ■
           ROR   BH, 1
           ROR   BH, 1                ;wyznacz w BX przesuniecie do
           AND   BX, 6000H            ;poczatku wiersza  o danym  Y
           MOV   AX, DX               ;współrzędna Y do rejestru AX
           SHR   AX, 1                ;dziel AX przez 4 -wynik ■ AL
           SHR   AX, 1
           MOV   AH, 90
           MUL   AH                   ;mnoż AL przez AH -wynik ■ AX
           ADD   BX, AX
           MOV   AX, CX               ;współrzędna X do rejestru AX
           SHR   AX, 1                ;podziel zawartosc AX przez 8
           SHR   AX, 1
           SHR   AX, 1                ;podziel zawartosc AX przez 8
           ADD   BX, AX               ;efektywne przesuniecie  w BX
           MOV   AL, ES: [BX]         ;pobierz bajt z pamięci do AL
           RET

Point_XY:  PUSH BX                    ;testuje punkt   (X=CX,  Y=DX)
           CALL  Laduj_XY             ;umiesc ■ AL bajt pam.  ekranu
           MOV   BX, CX
           AND   BX, 07H              ;operacja modulo 8  w rej. BX
           AND   AL, BYTE PTR MaskaBitu[BX]     ;wyodrebnij bit
           POP   BX
           JZ    Bit_EQ_0             ;jesli bit=0, ■ AL zostanie 0
           MOV   AL, 1                ;w przeciwnym razie  AL :=  1
Bit_EQ_0:  RET

Maskabitu: ■■    10000000B            ;tablica masek  poszcz. bitow
           ■■    01000000B
           DB    00100000B
           DB    00010000B
           ■■    00001000B
           DB    00000100B
           ■■    00000010B
           DB    00000001B

Ustal_StrG:MOV   AX, WORD PTR Nr_StronyG
           MOV   ES, AX
           RET


           ASSUME  DS:Kod_prog
Zainstaluj:MOV   SI, OFFSET Parametry
Instal_P:  LODSB                      ;nast. bajt bl. param. do  AL
           CMP   AL, Kod_CR           ;czy koniec bloku parametrów?
           JE    Kontyn_I             ;tak! nie ma juz wiecej opcji
           CMP   AL, '/'              ;czy jest to  zwiastun opcji?
           JNE   Instal_P             ;nie!  testuj  nastepny  znak
           LODSB                      ;nast. bajt bl. param. do  AL
           CMP   AL, Kod_CR           ;czy koniec bloku parametrów?
           JE    Kontyn_I             ;tak! nie ma już wiecej opcji
           CMP   AL, 'c'              ;czy jest to litera 'małe C'?
           JE    Tryb_Czek            ;tak - włącz tryb oczekiwania
           CMP   AL, 'C'              ;czy jest to litera 'duże C'?
           JE    Tryb_Czek            ;tak - włącz tryb oczekiwania
           CMP   AL, 't'              ;czy jest to litera 'małe t'?
           JE    Badaj_Tryb           ;tak -  włącz  badanie  trybu
           CMP   AL, 'T'              ;czy jest to litera 'duże T'?
           JE    Badaj_Tryb           ;tak -  włacz  badanie  trybu
           CMP   AL, '1'              ;czy  znak  jest  cyfrą '1' ?
           JE    WektorB800           ;tak -uaktywnij stronicę nr 1
           CMP   AL, '0'              ;czy  znak  jest  cyfrą '0' ?
           JNE   Instal_P             ;nie -pomin ją i testuj dalej
           MOV   WORD PTR Nr_StronyG, Str_gr_0   ;włącz strone 1
           JMP   Instal_P             ;i   testuj   nastepne  znaki

WektorB800:MOV   WORD PTR Nr_StronyG,  Str_gr_1
           JMP   Instal_P
Badaj_Tryb:MOV   WORD PTR Flaga_Tryb, 1
           JMP   Instal_P
Tryb_Czek: MOV   WORD PTR Flaga_Czek, 1


           JMP   Instal_P
Kontyn_I:  MOV   AH, CzytajWekt
           MOV   AL, PrtSc_Wekt
           INT   MS_DOS               ;ES:BX=wektor przerwania nr 5
           MOV   WORD PTR PrtSc_Adr, BX ;pam. offset przerwania
           MOV   WORD PTR PrtSc_Segm, ES ;i   segment przerwania
           MOV   DX, OFFSET Herkules
           MOV   AH, Ustaw_Wekt
           MOV   AL, PrtSc_Wekt       ;DS:DX=nowy wektor przerwania
           INT   MS_DOS               ;zapisz  nowy  wektor obsługi
           MOV   DX, OFFSET Zainstaluj
           INT   Zintegruj            ;niech program rezyduje w PAO
Kod_prog   ENDS

Dane_DOS   SEGMENT AT 40H
           ORG    256
PrtSc_Stat LABEL BYTE                  ;bajt statusu operacji PrtSc
Dane_DOS   ENDS
           END       Inicjacja
```

LOADER PROGRAMU W BASICU

```
10  [illegible] Program ładujący kod HERCGRAK.   Roland Wacławek 1987
20  OPEN "O", #1, "HERCGRAK.COM"
30  SUMA= 0
40  FOR NR.LINII=200 TO 470 STEP 10
50     READ LINIA$
60     FOR  NR.BAJTU=1 TO 24
70        BAJT= VAL("&H"+MID$(LINIA$, NR.BAJTU*2-1, 2))
80        SUMA=SUMA+BAJT
90        PRINT #1, CHR$(BAJT);
100    NEXT NR.BAJTU
110 NEXT NR.LINII
120 IF SUMA<>72882! THEN PRINT "Blad w bloku DATA !":  STOP
130 CLOSE #1
140 END
200 DATA "E93702000000000000000B8000000000000FB061E52515350"
210 DATA "B840008EC026803E0001017503E9A3010E1FB402CD16A20E"
220 DATA "01F6060C01FF7413B40FCD103C06740B585B595A1F072EFF"
230 DATA "2E0301F6060D01FF7429B40EB007CD10B400CD163C1B74C5"
240 DATA "BA00B03D003B740FBA00B83D003C74073D003D74CBEBE189"
250 DATA "16090189260701B018E85E01E8AD01A00E0124017403E98A"
260 DATA "0033D2B9C80252E858010AC0751D42F6C20775F383EA0883"
270 DATA "E90879EBE823015A83C20881FA5D0172DAEB7E5A5283C108"
280 DATA "890E0F018BC1E8AE0033C981FA54017721 33DB52E84101D0"
290 DATA "C30AD842FEC780FF0875F15A8AC3E8BB00413B0E0F0175E1"
300 DATA "EBBA33DB52E82001D0C30AD842FEC780FF0475F15A8AC3D0"
310 DATA "E0D0E0D0E0D0E0E892004 13B0E0F0175D9EB9133C933D2E8"
320 DATA "D0000AC075184281FA5D0175F2E8A20083C10881F9D00275"
330 DATA "E432F6EB7D90B85D012BC28BD803C003C003C3E822008916"
340 DATA "0F01BA5C01E89A00E84900E84600E84300E84000E83D003B"
350 DATA "160F0174C04AEBE55150B95A00EB045150B14CB01BE82400"
360 DATA "8AC1E81F00598AC1E819008AC5E8140059C3B401CD16741B"
370 DATA "B400CD163C1B75F2B600EB125250B40033D2CD17F6C42575"
380 DATA "03585AC3B6FF8B260701B840008EC0B82400E81D00FA2688"
390 DATA "360001585B595A1F07CF50B00DE8CCFFB00AE8C7FFE8B2FF"
400 DATA "58C350B01BE8BCFFB033E8B7FF58E8B3FFC38AFAD0CFD0CF"
410 DATA "D0CF81E300608BC2D1E8D1E8B45AF6E403D88BC1D1E8D1E8"
420 DATA "D1E803D8268A07C353E8D6FF8BD981E30700228 72C035B74"
430 DATA "02B001C3804020100804020 1A109018EC0C3BE8100AC3C0D"
440 DATA "74413C2F75F7AC3C0D74383C63742C3C43742 83C74741C3C"
450 DATA "6474183C31740C3C3075DAC706090100B0EBD2C706090100"
460 DATA "B8EBCAC7060C010100EBC2C7060D010100EBBAB435B005CD"
470 DATA "21891E03018C060501BA1101B425B005CD21BA3A03CD2700"
```

Aby uzyskać działający program, należy wprowadzić jego postać źródłową, korzystając z dowolnego edytora zapisującego teksty bez znaków formatujących: EDLIN, SideKick, edytor TURBO-Pascala itd. Komentarze można oczywiście pominąć. Program źródłowy należy zapisać np. w pliku HERCGRAK. ASM. Następnie wywołujemy makroasembler zleceniem: MASM HERCGRAK. Cztery kolejne pytania makroasemblera kwitujemy naciśnięciem klawisza **[Enter]**. Jeśli wystąpią jakiekolwiek błędy, należy porównać wkazane na ekranie instrukcje z tekstem źródłowym. Jeśli asembler nie wykrył błędów, wywołujemy linker zleceniem: LINK HERCGRAK, pytania kwitujemy klawiszem **[Enter]**.

Ostrzeżenie o braku segmentu stosu jest w tym przypadku bez znaczenia. Ostatni etap to przetworzenie programu z formatu EXE na COM. W tym celu należy podać zlecenie: EXE2BIN HERCGRAK. EXE HERCGRAK.COM.

Program HERCGRAK może posłużyć za punkt wyjścia do własnych eksperymentów nad rozszerzaniem możliwości IBM-PC/XT/AT.

COMMODORE PC I

Modnym tematem i przedmiotem pożądania wielu stał się dziś, chyba jednak nieco przereklamowany, mikrokomputer Amstrad PC 1512. Warto więc wiedzieć, że firma Amstrad nie jest dziś jedynym producentem komputerów domowych, kompatybilnych z IBM PC. Co więcej, jej próba ekspansji na rynek USA zapewne zakończy się sromotną porażką. Przyczyna jest prosta: potężni i bardziej zasiedziali na amerykańskim rynku rywale, firmy Atari i Commodore przedstawiły własne konstrukcje komputerów osobistych, kompatybilnych z IBM PC. Dziś przedstawiamy Commodore PC I, który jest dostępny także w Europie.

W odróżnieniu od firmy Amstrad, konstruktorzy firmy Commodore poświęcili wzrost wydajności na rzecz maksymalnej kompatybilności z IBM PC. Zastosowano procesor 8088 z częstotliwością zegarową 4.77 MHz. Podobnie jest w PC 1512, dzięki zastosowaniu specjalizowanych układów scalonych wielkiej skali integracji udało się ograniczyć liczbę elementów i umieścić na wspólnej płycie właściwy mikrokomputer, sterownik wizyjny, kontroler dysków elastycznych i interfejsy: szeregowy i równoległy.

Pamięć RAM ma pojemność 512 KB i może być w prosty sposób powiększona do standardowych 640 KB. Stěrownik graficzny odpowiada funkcjonalnie CGA i może współpracować z dowolnym, standardowym monitorem jedno- lub wielobarwnym. Zasilacz mieści się w płaskiej i zwartej obudowie komputera o rozmiarach 33 × × 32 × 8,5 cm. W obudowie znalazło się jeszcze miejsce dla jednej stacji dysków elastycznych 360 KB. Następne stacje można dołączać z zewnątrz, co nie jest niestety rozwiązaniem najszczęśliwszym. To samo dotyczy ewentualnych, dodatkowych pakietów, które muszą być instalowane w zewnętrznej kasecie — w środku brak miejsca (magistrala komputera jest wyprowadzona na zewnątrz).

Klawiatura komputera zasługuje na pochwałę — pod tym względem PC I dystansuje swego rywala PC 1512. Jest ona płaska i ergonomiczna, zaś jej układ przypomina nieco klawiaturę PC/AT z dużymi klawiszami Shift i Enter. Klawisze: CapsLock i NumLock mają lampki, sygnalizujące ich stan.

Przedstawiona koncepcja wskazuje, że PC I ma być komputerem o charakterze zdecydowanie domowym i edukacyjnym, przeznaczonym dla użytkowników szukających taniego sprzętu o wysokich walorach użytkowych i bogatym oprogramowaniu. Ostateczna cena PC I nie była jeszcze znana w chwili składania niniejszego materiału w redakcji. Można jednak przypuszczać, że Commodore nie będzie dla firmy Amstrad łatwym przeciwnikiem... **(rw)**



CeBIT '87
— NOWOŚCI W SPRZĘCIE MIKROKOMPUTEROWYM



Atari PC — najnowsza maszyna koncernu Jacka Tramiela. W założeniach prosty konstrukcyjnie model zgodny ze standardem IBM PC — powinien on za jakiś czas być jedną z najtańszych propozycji w tej grupie. Jednostka centralna umieszczona jest w płaskiej obudowie, na niej ustawiony jest moduł rozszerzenia zawierający dodatkową stację dysków oraz Winchester



A oto nieco starszy model europejskiego komputera kompatybilnego z IBM PC — Schneider PC. Maszyna ta cieszy się dużym uznaniem profesjonalistów, popyt na nią nie maleje. Na fotografii model wyposażony w Winchester 20 MB

W jednym z numerów „Młodego Technika" opisaliśmy wrażenia z targów CeBIT'87 — dotyczyły one ogólnych tendencji rozwoju techniki biurowej, automatyzacji produkcji i sprzętu z tym związanego. Nie napisaliśmy jednak nic bliższego o sprzęcie mikrokomputerowym klasy nieco niższej, niż stricte profesjonalny oraz o ciekawych urządzeniach peryferyjnych. Temu poświęcony jest niniejszy odcinek relacji. Analizując przedstawione poniżej informacje można odnieść wrażenie, że nastąpiło jakby spowolnienie postępu w zakresie mikrokomputerów i urządzeń peryferyjnych dla szerokiego kręgu odbiorców. Jest to chyba częściowo prawda — nie zapominajmy jednak, że im wyższy poziom techniki, tym mniej gwałtowny postęp przeradzający się nieraz w może dość powolną, ale stabilnie kroczącą ewolucję zmierzającą do ciągłego doskonalenia wytworów cywilizacji.

Przegląd ciekawostek rozpoczniemy od sfery mikrokomputerów domowych. Dziedzina ta, aczkolwiek nadal niezwykle interesująca, zaczyna jednak coraz bardziej ustępować miejsca komputerom osobistym. Nie pomaga już stosowanie 16-bitowych mikroprocesorów oraz ciągłe powiększanie mocy obliczeniowych (nierzadko przewyższających wydajność typowego IBM PC!). Rynek jest już bardzo mocno nasycony komputerowymi „zabawkami" i przeciętny użytkownik szuka dziś sprzętu pomagającego także w pracy. Tu dochodzi do głosu oprogramowanie — wygrywa ten standard, który oferuje najwięcej możliwości, ma najbogatszą bibliotekę programów, jest tani w eksploatacji. Aż się prosi więc IBM PC...

Wiele firm broni się przed widmem utraty rynku poprzez wprowadzenie modeli zgodnych z IBM — zrobił to już dość dawno Commodore, nieco później Schneider. Ostatnio do grupy tej dołączyło nawet Atari prezentując swój nowy model ATARI PC. Kilka danych technicznych tego komputera: procesor 8088 z zegarem 4,77/8,0 MHz (przełączany), podstawka dla koprocesora arytmetycznego, standardowo 512 KB RAM z możliwością rozszerzenia do 640 KB, wydzielona pamięć obrazu 256 KB, wbudowana stacja dysków 5.25

o pojemności 360 KB, wbudowany interfejs równoległy, szeregowy oraz myszy, tryby graficzne zgodne z kartami CGA, EGA oraz Hercules (przełączane). A więc maszyna ta wyposażona jest standardowo we wszystkie podstawowe elementy IBM. Ma ona kosztować łącznie z monitorem poniżej 500 dolarów. Tak niską cenę wyrobu osiągnięto przez maksymalne ograniczenie zbędnych (zdaniem twórców) elementów. Tak więc ATARI PC nie posiada typowych dla IMB złącz (slotów) dla zainstalowania dodatkowych kart — obudowa komputera jest bardzo płaska i mieści w środku oprócz zasilacza jedynie główną płytę logiki, podstawową jednostkę dysków 5,25 cala oraz dodatkowo drugą taką jednostkę albo dysk twardy Winchester. Dalsze możliwości rozbudowy możliwe są jedynie przez wykorzystanie dodatkowej obudowy, ustawianej na głównej. Ogranicza to znacznie możliwości zastosowania nowego ATARI, ale być może rzeczywiście korzystna cena spowoduje, że będzie to atrakcyjny sprzęt dla „domowych" profesjonalistów.



Macintosh II — jeszcze jeden nowy komputer o otwartej architekturze i bardzo dużej mocy obliczeniowej (32-bitowy procesor, koprocesor arytmetyczny, min. 1 MB RAM). Jest to jeszcze jeden przykład na to, że duże wydajne maszyny muszą mieć możliwość rozbudowy o różne specjalizowane układy

Dość popularny już komputer ATARI ST także doczekał się nowej wersji:ATARI MEGA ST posiada już 1, 2 lub 4 MB pamięci operacyjnej, ale przede wszystkim umieszczony jest w nowej obudowie, oddzielonej od klawiatury. Przypomina to nieco konstrukcję IBM, gdzie bodajże po raz pierwszy jednostkę centralną wraz z zasilaczem zabudowano do obudowy spełniającej rolę podstawy monitora. Zaletą nowej koncepcji jest większa wygoda obsługi (oddzielna klawiatura), lepsze chłodzenie części elektronicznej i łatwiejsze podłączanie modułów rozszerzających możliwości komputera — umieszczone są one w identycznej obudowie, jak jednostka centralna, postawionej na niej piętrowo (tak samo, jak w przypadku ATARI PC). W chwili obecnej dostępny jest dysk twardy 20 MB ATARI SH 205. Dużą jego zaletą jest komunikacja z komputerem przez bardzo szybki kanał DMA.

Komputery ATARI ST zdobyły rynek bardzo korzystnym stosunkiem ceny do możliwości sprzętowych i programowych. Proporcje te są ciągle poprawiane i prawdopodobnie dlatego ATARI ST są jednymi z najlepiej sprzedawanych komputerów klasy jeszcze domowej (chociaż możliwości jego są porównywalne z klasą komputerów osobistych) — w roku ubiegłym tylko w RFN sprzedano ich ponad 73 tysiące.

Dobitnym przykładem polityki prowadzonej przez firmę ATARI jest opracowanie ciekawej konstrukcyjnie drukarki laserowej ATARI SLM. Parametry techniczne są praktycznie porównywalne z innymi drukarkami laserowymi (rozdzielczość 300 dpi — punktów na cal — szybkość: 8 stron na minutę, format A4), co wynika prawdopodobnie z zastosowania standardowego mechanizmu drukującego. Różnica w stosunku do innych konstrukcji tego typu polega na znacznym ograniczeniu części sterującej. Typowo zawiera ona szybki mikroprocesor (na ogół Motorola 68000), dużą pamięć operacyjną rzędu 1 MB, dużą pamięć stałą z kilkoma zestawami znaków, krojami pisma itp. ATARI SLM posiada jedynie część sterującą bezpośrednio drukiem oraz szybki układ DMA — za jego pomocą drukarka pobiera dane do druku bezpośrednio z pamięci komputera ATARI ST. Wszystkie funkcje wyznaczania matrycy znaków, komponowania elementów graficznych itd. przejmuje komputer. Odpowiedni program graficzny zastępuje więc wewnętrzne oprogramowanie drukarki. Jest to bardzo wygodne, gdyż generatory znaków można każdorazowo załadować z dyskietki. Bardzo łatwe jest też łączenie elementów graficznych z tekstem, a ponadto obraz przyszłego wydruku widoczny jest cały czas na ekranie. Oczywiście pewnym ograniczeniem jest możliwość dołączenia drukarki tylko do ATARI MEGA ST, lecz zważywszy niższą od konkurencyjnych cenę samej drukarki, dla posiadaczy tego komputera jest to dość atrakcyjna propozycja.

Podobną politykę do ATARI wyznaje znana u nas firma Schneider (jej partnerem jest Amstrad). Jej dwa szlagiery — Joyce i PC 1512 — odniosły na rynku zachodnim dość duży sukces, głównie dzięki stosunkowo niskiej cenie i dobrej jakości. Firma na tegorocznych targach w zasadzie nie zaprezentowała jakichś nowych opracowań, a jedynie rozwinięcie wcześniej znanych modeli (przykładowo Joyce Plus, PCW 8512, wyposażony w pamięć 512 KB i drugą stację dysków o pojemności 1 MB). Ciekawą ofertą były trzy drukarki: DMP2000, 9-igłowa o charakterystycznym poziomym przesuwie papieru, drukująca w trybie normalnym z szybkością 105 znaków na sekundę, jedna z najtańszych, w swojej klasie (tylko 600 DM!).

Model DMP4000, to już drukarka profesjonalna, wykorzystywana m.in. w konfiguracji ze Schneiderem PC. Szybkość druku — 200 zn/s plus oczywiście NLQ i inne możliwości typowe dla drukarek tej klasy. Najlepszy model drukarki Schneidera — 24-igłowa SD24 — to już urządzenie najwyższej klasy. W trybie normalnym drukuje z szybkością 135 znaków na sekundę (ten tryb bardziej przypomina NLQ zwykłych drukarek, niż tzw. draft!), w trybie korespondencyjnym (LQ) — 54 zn/s. Cena drukarki — ok. 1300 DM.

Firma APPLE znana jest z komputerów wysokiej jakości przeznaczonych do poważniejszych zastosowań domowych. Maszyny z serii Macintosh są już bardzo wydajnym narzędziem, nawet do zastosowań czysto profesjonalnych. Na targach zaprezentowano dwa nowe modele tego komputera. Pierwszy, Macintosh SE, na pierwszy rzut oka niewiele się różni od swojego poprzednika Macintosha Plus. Różnica kryje się we wnętrzu komputera — miejsce na przyłączenie specjalnej karty komunikacyjnej. Umożliwia ona za pomocą tzw. magistrali Macintosh SE dołączenie innych urządzeń takich, jak np. większe terminale graficzne, łącze z innymi komputerami itp. Zastosowane nowe



AMIGA 2000 — rozszerzenie znanego już sprzed roku modelu AMIGI 1000. Komputer ten posiada znaczne możliwości rozbudowy, po zamontowaniu specjalnej karty IMB pracuje także w systemie PC DOS (komputer akceptuje wtedy oprogramowanie IMB PC/XT lub AT)

układy scalone wielkiej skali integracji zwiększyły szybkość pracy komputera o ok. 15—20% względem poprzednika. Wbudowany interfejs SCSI umożliwia dołączenie maksymalnie do 6 urządzeń zewnętrznych.

Drugi model Macintosha — oznaczony symbolem II — jest już komputerem o otwartej architekturze. Posiada 32-bitowy procesor Motorola 68020 (zegar 15,67 MHz), koprocesor 68881, pamięć standardowo 1 MB RAM (wewnętrznie rozszerzalna do 8 MB, zewnętrznie do 2 GB), 256 KB ROM. Główna różnica w stosunku do poprzedników polega na wydzieleniu na głównej płycie 6 miejsc (slotów) na dodatkowe karty, rozszerzające możliwości systemu. Jest to zatem filozofia podobna do wykorzystywanej przez IBM PC. Bogaty zestaw standardowych interfejsów w połączeniu z możliwościami rozszerzenia struktury systemu otwiera dla maszyny tej nowe, poważniejsze zastosowania, głównie w zakresie automatyzacji prac biurowych, sieci przesyłu danych, systemów CAD/CAM i wiele innych.



AMIGA 500 — nowy szlagier Commodore. Mikrokomputer ten jest programowo niemal zupełnie zgodny z AMIGĄ 1000, a przy tym kosztuje tyle, ile komputer klasy C-128D czy ATARI ST. Czy to wystarczy do zdobycia rynku, zobaczymy już niebawem. W tyle nowa, 24-igłowa drukarka MPS2000

Ciekawą propozycją jest zewnętrzna jednostka dysków 5,25 cala oraz karta PC dla Macintosha II — za ich pomocą możliwa jest praca pod systemem MS DOS, czyli wykorzystanie bardzo wielu programów przeznaczonych dla IBM PC. Podnosi to znacznie walory użytkowe Macintosha II, ale podkreśla też znaczenie standardu „big blue". Nie wdając się w polemikę na temat, który komputer jest lepszy, można z całą pewnością stwierdzić, że połączenie świata IBM ze światem Macintosha dzięki odpowiednim rozwiązaniom sprzętowym z pewnością przyniesie duże korzyści obu stronom.

Na koniec prezentacji nowości przedstawionych przez producentów komputerów domowych zostawiliśmy Commodore. Firma ta, po okresie pewnego zastoju, wyraźnie ruszyła się do przodu. Z ciekawych propozycji należy wymienić nowy model zgodny z IBM — Commodore PC I. Jest to konstrukcja o zminimalizowanej ilości elementów — dzięki wykorzystaniu układów scalonych wielkiej skali integracji zaoszczędzono ok. 45 elementów.

Wszystkie podstawowe elementy systemu umieszczone są na jednej płycie, także sterownik wizyjny o rozdzielczości 320×200 punktów w 4 kolorach lub 640×200 w dwóch. Część elektroniczna wraz z zasilaczem i jednostką dysków elastycznych 5,25 cala umieszczona jest w płaskiej obudowie, na której można ustawić monitor. Wyrób ten w czasie targów jeszcze nie był w sprzedaży, ale oczekuje się, że będzie on stosunkowo tani. Spodziewać się należy konkurencji ze strony ATARI PC — zobaczymy, kto wygra tę walkę.

Największym jednak zainteresowaniem cieszyły się dwa nowe modele AMIGI: AMIGA 2000 i AMIGA 500. Pierwszy z nich, to system o otwartej architekturze, bazujący na strukturze poprzednika, AMIGI 1000. Rozbudowy systemu można dokonać wykorzystując następujące złącza:

— jedno złącze karty procesora systemowego (np. Motorola 68020/68881 — tzw. Turbo-karta, czyniąca wydajność komputera znacznie wyższą),
— 5 złącz kart podłączonych do magistrali AMIGI,
— złącze adaptera IBM PC/XT lub AT,
— 3 złącza kart XT lub AT.

Standardowo AMIGA 2000 wyposażona jest w 1 MB RAM, 256 KB ROM, dysk elastyczny 3,5 cala 880 KB, opcjonalnie dysk elastyczny lub dysk twardy 3,5 cala lub 5,25 cala (standard IBM). Maksymalnie pamięć można rozszerzyć wewnętrznie do 9,5 MB, możliwe jest oczywiście dołączenie dodatkowych pamięci zewnętrznych. Komputer ten w wersji bez monitora i z jednym napędem 3,5″ kosztuje obecnie nieco poniżej 3000 DM.

Bardzo interesująco przedstawiają się perspektywy połączenia ciekawych możliwości graficznych i muzycznych AMIGI z całą paletą funkcji oferowanych przez system IBM PC/XT lub AT.

Dla „domowych" hobbistów mikrokomputerowych niewątpliwie najciekawszą propozycją spośród wszystkich pokazanych na tegorocznym Cebicie jest AMIGA 500. Komputer ten reklamuje się jako pierwsza maszyna 16/32-bitowa do użytku domowego, o wielozadaniowym systemie operacyjnym. Brzmi to rzeczywiście obiecująco, a zważywszy cenę (bez monitora) około 1250 DM można spodziewać się dużej popularności tego komputera. Czym jest AMIGA 500 od środka? Jest to nic innego, jak nieco „uprzątnięta" AMIGA 1000 — kilkadziesiąt układów scalonych małej i średniej skali integracji zastąpiono kilkoma wyspecjalizowanymi LSI. Struktura pozostała jednak bez zmian — a zatem w skrócie: procesor Motorola 68000, trzy specjalizowane układy VLSI do obsługi grafiki, dźwięku i zarządzania dostępem do pamięci, standardowo 512 KB RAM (możliwość rozszerzenia do 1 MB za pomocą niewielkiego dodatkowego modułu zawierającego także zegar z podtrzymaniem bateryjnym), 256 KB ROM, jednostka dysków 3,5" o pojemności 880 KB (po sformatowaniu) plus ew. dodatkowa jednostka zewnętrzna, interfejs szeregowy i równoległy, przyłącza myszy, pióra świetlnego i innych urządzeń. Pozostają także znane z AMIGI 1000 cztery niezależne kanały dźwiękowe obsługiwane przez przetworniki D/A oraz fantastyczna grafika. System operacyjny AMIGA DOS, wielozadaniowy.

Całość, a zatem główna płyta logiki, napęd dysków oraz klawiatura i zasilacz umieszczone są w jednej obudowie, przypominającej nieco kształtem C128. Monitor dołączamy przez wyjście RGB, dla telewizora kolorowego wymagany jest niestety zewnętrzny koder PAL.

Cenowo komputer ten jest bardzo zbliżony do C128D, posiada już bardzo bogate oprogramowanie (akceptuje niemal cały zbiór programów na AMIGĘ 1000), jest wygodny w użytkowaniu. Wróży mu to szybką karierę, ale na razie jest to zupełna nowość. Być może znajdzie on zastosowanie w szkołach, gdzie można znakomicie wykorzystać jego możliwości przetwarzania i analizy obrazów.

Z innych nowości przedstawionych przez COMMODORE na targach należy jeszcze wymienić nowy model kompatybilny z IBM PC AT — COMMODORE PC 40/40, półprofesjonalny Winchester 20 MB dla C64, rozszerzenie pamięci C64 do 256 KB RAM oraz nową 24-igłową drukarkę MPS 2000 (drukuje w kolorze).

Bogumił Lotaz

PIERWSZY KOMPUTER OSOBISTY Z PROCESOREM 32-BITOWYM

Amerykańska firma Compaq specjalizuje się w produkcji komputerów kompatybilnych z wyrobami IBM. Jej dewizą jest przy tym nie tyle niewolnicze naśladownictwo oryginału, lecz jak najdalej idące ulepszenia pierwotnej koncepcji. Tym razem Compaq podjęła jednak bardziej zdecydowaną próbę zaakcentowania swej technologicznej wyższości nad IBM. Okazję ku temu stworzyło niedawne pojawienie się nowego mikroprocesora INTEL 80386.

INTEL 80386 cechuje się konsekwentną architekturą 32-bitową, możliwością adresacji do 4 GB PAO i dużą wydajnością, a przy tym jest zgodny programowo z 8088/6 oraz 80286. Procesor ten jest sercem nowego mikrokomputera COMPAQ Deskpro 386. Częstotliwość zegarowa wynosi 16 MHz (procesor pracuje w tempie 4 MIPS), co zapewnia komputerowi 2 do 3 razy większą wydajność od IBMAT z częstotliwością zegarową 8 MHz. Istnieje możliwość zainstalowania koprocesora arytmetycznego. Ponieważ jednak koprocesora INTEL 80387 nie ma jeszcze na rynku, zastosowano 80286, przeznaczony pierwotnie do współpracy z 80286.

Pamięć RAM liczy 1 MB, można ją jednak rozbudować do 10 MB na płycie głównej, a przy użyciu dodatkowej karty pamięci — do 14 MB. Jako pamięć masowa występuje stacja dysków elastycznych 1.2 MB (jak w AT) oraz dysk Winchester o pojemności 40 lub 130 MB. Współpracujący z barwnym monitorem kontroler graficzny odpowiada standardowi karty EGA, może też emulować popularną kartę grafiki barwnej CGA i monochromatyczną HERCULES.

Nie spotykana dotąd w komputerach osobistych wydajność predestynuje COMPAQ 386 do takich zastosowań jak komputerowe wspomaganie projektowania (CAD), sztuczna inteligencja oraz inżynieria oprogramowania. COMPAQ 386 jest zgodny programowo z IBM PC/AT i może korzystać praktycznie z całego oprogramowania dostępnego w systemie MS-DOS.

Mimo nowoczesnego procesora trudno uznać jednak COMPAQ 386 za prawdziwą rewolucję. Wskazuje na to już fakt zastosowania w nim systemu MS-DOS 3.1. Potrzeba zachowania zgodności z istniejącym oprogramowaniem uniemożliwia bowiem pełne wykorzystanie procesora 80386. Gospodarka pamięcią jest taka, jak w IMB AT. Procesor pracujący z systemem operacyjnym MS-DOS (PC-DOS) emuluje mikroprocesor 8086, a więc główny zysk z jego zastosowania polega na wyższej wydajności. Brak też trybu pracy, w którym byłyby wykorzystane wszystkie możliwości 80386, m.in. pamięć wirtualna. Mankamenty te stanowią jednak cenę, jaką COMPAQ musiała zapłacić z powodu praktycznego braku oprogramowania, wykorzystującego atuty nowej rodziny procesorów 80286/386. Dość powiedzieć, że w IBM AT zaawansowane możliwości procesora także są niewykorzystane — także i tam emuluje on głównie 8086.

Sytuacja w zakresie oprogramowania wykorzystującego możliwości nowych procesorów marki INTEL ma szansę ulec w roku 1987 znacznej poprawie. Firma MICROSOFT zapowiada bowiem nową wersję systemu operacyjnego MS-DOS 5,0. Tym razem zdecydowano się zerwać z hamującą postęp techniczny zasadą ścisłej kompatybilności z wersjami poprzednimi. MS-DOS 5.0 nie będzie więc pracował na komputerach z procesorami 8088/6, lecz jedynie na IBM AT i podobnym sprzęcie z procesorami 80286 i 386. Zniknie wymuszane przez procesory 8088/86 ograniczenie dostępnego dla systemu obszaru pamięci do 1 MB. Nowy system operacyjny będzie cechował się nowymi zasadami gospodarki pamięcią (przydział PAO na żądanie zamiast sztywnego podziału pamięci podczas ładowania programów), oraz wielozadaniowością. Jak widać, komputery osobiste coraz bardziej upodabniają się do większych systemów — nie tylko pod względem wydajności i pojemności pamięci, ale i filozofią oprogramowania.

Dużo uwagi poświęcono skutecznej separacji równoległych procesorów i zabezpieczeniu przed ich wzajemnym oddziaływaniem oraz możliwości pracy w sieciach komputerowych. MS-DOS 5.0 będzie używać (nareszcie!) naturalnego (*protected mode*) trybu pracy 80286/386. Niestety, konstrukcja procesorów nie pozwala na łatwe przełączanie między trybem naturalnym a emulacją 8086. Programiści firmy MICROSOFT rozwiązali jednak ten problem — tym samym w nowym systemie operacyjnym będzie można nadal korzystać z obfitości oprogramowania „starych" wersji systemu. Od strony użytkownika obsługa ułatwiać ma znany już system MS-WINDOWS, wykorzystujący okienka ekranowe i rozwijane *menu* do ułatwienia dialogu człowieka z maszyną i równoległej pracy z kilkoma programami. MS-WINDOWS cechuje się podobną koncepcją, jak GEM firmy Digital Research, lecz wydaje się mieć większe możliwości i lepsze perspektywy. Tego typu systemy wymagają znacznej pamięci roboczej. MS-DOS 5.0 — nowa generacja mikrokomputerów z pamięcią o pojemności kilku-kilkunastu MB — stwarzają dla MS-WINDOWS znacznie dogodniejsze warunki pracy, eliminując konieczność przejściowego przechowywania fragmentów pamięci operacyjnej na dysku, w celu udostępnienia jej dla innych zastosowań.

Roland Wacławek



Jak z ZX SPECTRUM zrobić komputer?

ZMIENIAMY ARCHITEKTURĘ KOMPUTERA

Wojciech Apel

Artykuł ten stanowi kontynuację tematu udoskonalenia mikrokomputera ZX Spectrum rozpoczętego w numerze 1/1987 naszego kwartalnika. Proponowane w niniejszym artykule przeróbki mikrokomputera ZX Spectrum mają na celu zmianę stosunkowo sztywnej struktury sprzętu. Polegają one na dołączeniu dodatkowych stron (bloków) pamięci tak, aby konfiguracja tego systemu dała się łatwo zaadaptować do dowolnego układu.

Przypatrzmy się mapie pamięci ZX Spectrum. Od adresów 0 do 3FFFH procesor „widzi" pamięć ROM zawierającą rezydujący tam interpreter języka BASIC. Powyżej, tj. od 4000H do 7FFFH zlokalizowana jest pamięć RAM o pojemności 16 KB, w której znajduje się obraz ekranu, zmienne systemowe i rozwijająca się w górę treść programu w języku BASIC (rys. 1). Pamięć zawierającą treść ekranu nazwiemy VRAM (Video-RAM). Komputery 48K są wyposażone w opcjonalną pamięć RAM 32K umieszczoną w przestrzeni adresowej od 8000H do FFFFH (podobną strukturę stosuje się w komputerach MERITUM czy Laser). Struktura taka posiada jednak pewne wady.

Pierwsza z nich to wyróżnienie pewnego obszaru RAM, jako pamięci obrazu. Obszar ten należy do przestrzeni adresowej procesora, a więc umieszczanie w nim programów w języku wewnętrznym mikroprocesora jest możliwe. Jednak wtedy na ekranie pojawiają się tzw. śmieci nie będące żadną informacją dla użytkownika. Przykładem tego może być znany program kopiujący COPY-COPY.

Drugą wadą takiej struktury jest umiejscowienie pamięci ROM od adresu 0. Po sygnale RESET (np. po włączeniu zasilania) procesor startuje wykonując program od początku przestrzeni adresowej (tj. od adresu 0) — powinien więc tam znajdować się program uruchamiający komputer. W bardziej rozbudowanych rozwiązaniach stosuje się w tym miejscu małe pamięci ROM, w których znajduje się program ładujący system operacyjny z urządzenia zewnętrznego. Po załadowaniu i uruchomieniu systemu operacyjnego odłącza on tę małą pamięć i odsłania istniejącą pod nim pamięć RAM. Ta mała pamięć ROM nosi nazwę BOOTSTRAP. Rozwiązanie takie stosuje się np. w mikrokomputerach pracujących pod kontrolą systemu operacyjnego CP/M. System ten wykorzystuje komórki od 0 do 255, jako zmienne systemowe i posiada możliwość definiowania *restartów* dla programów uruchamianych pod jego kontrolą — konieczna jest więc pamięć RAM.

Proponowane niżej przeróbki Spectrum mają na celu usunięcie najistotniejszych wad jego struktury, co pozwoli na pracę pod kontrolą systemu CP/M. Oto wymagania stawiane mikrokomputerowi po przeróbce:

1. Wszystkie modyfikacje nie mogą zmienić struktury mikrokomputera w sposób istotny, tzn. musi istnieć możliwość pracy z normalnym interpreterem języka BASIC zapisanym w firmowej pamięci ROM.

2. Wszystkie programy napisane na mikrokomputer ZX Spectrum muszą pracować na przerobionym komputerze. Założenie to obejmuje także programy napisane w języku wewnętrznym komputera.

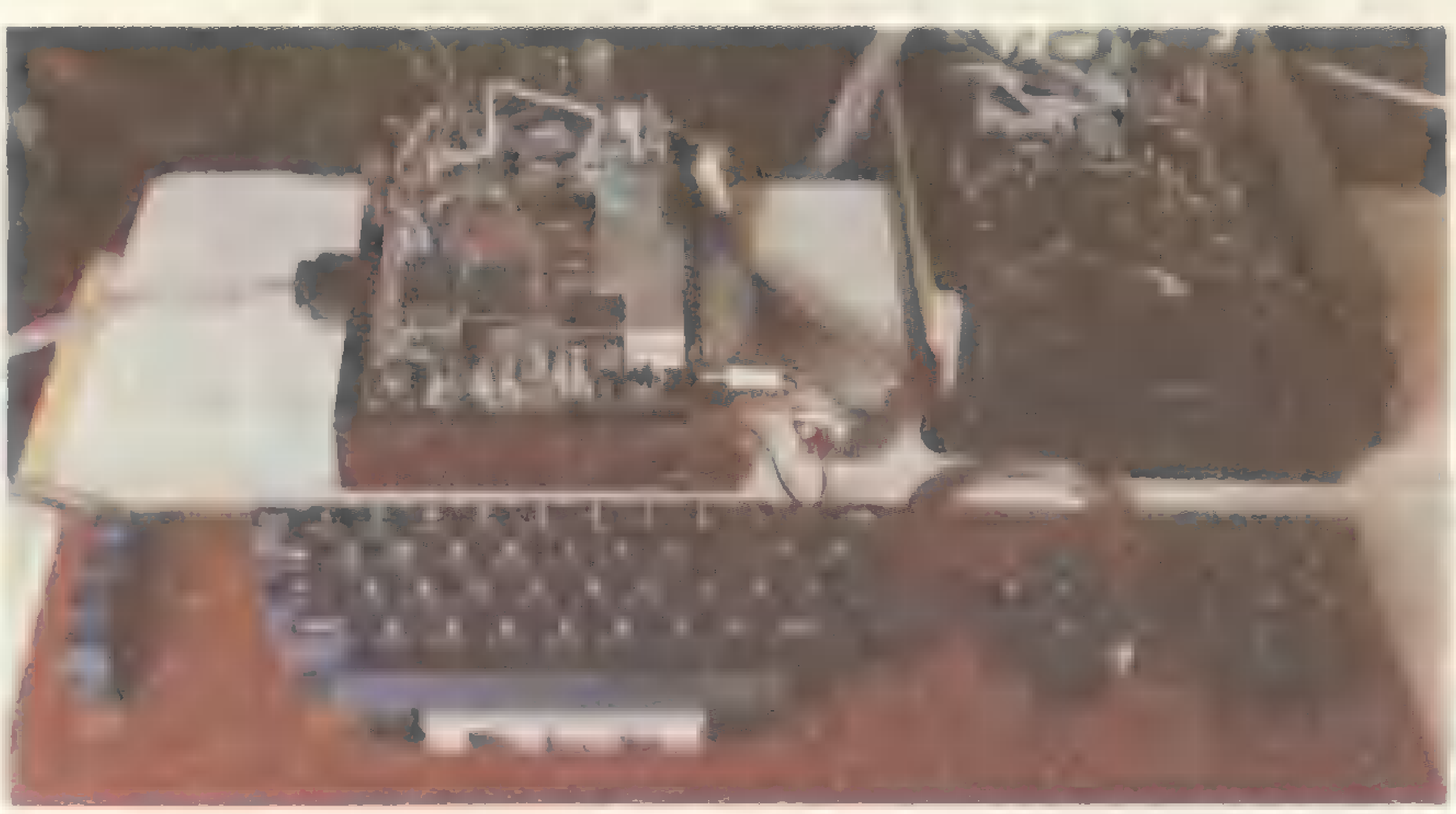
3. Wszystkie zmiany muszą zmieścić się w normalnej obudowie komputera i muszą być zasilane ze standardowego zasilacza.

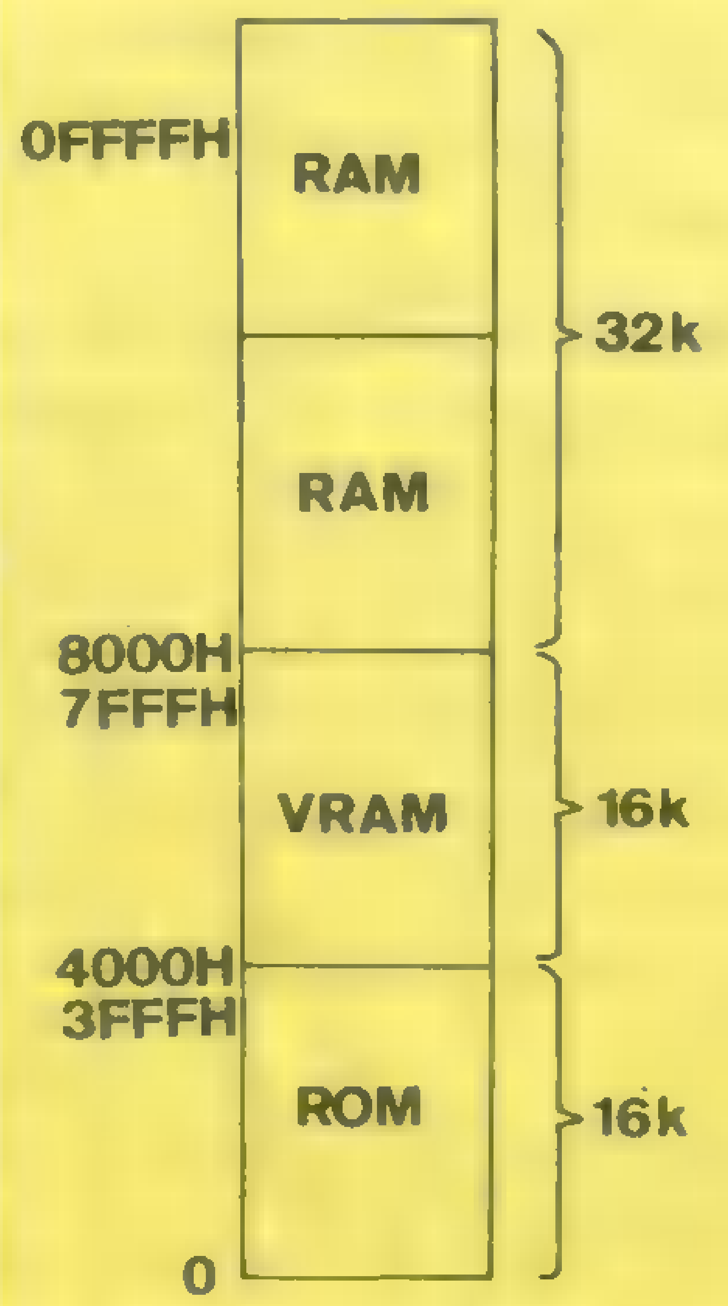
4. Wszystkie sygnały na złączu krawędziowym muszą być zgodne z standardowym ZX Spectrum. Gwarantuje to możliwość pracy ze wszystkimi interfejsami fabrycznymi i amatorskimi.

5. Możliwość pracy pod kontrolą systemu CP/M, lub wszystkich innych systemów bazujących na mikroprocesorze Z-80 (np. ISIS), a nie wymagających więcej, niż 64 KB pamięci RAM. Pozwala to wykorzystać dużą część bogatego oprogramowania pracującego pod tymi systemami.

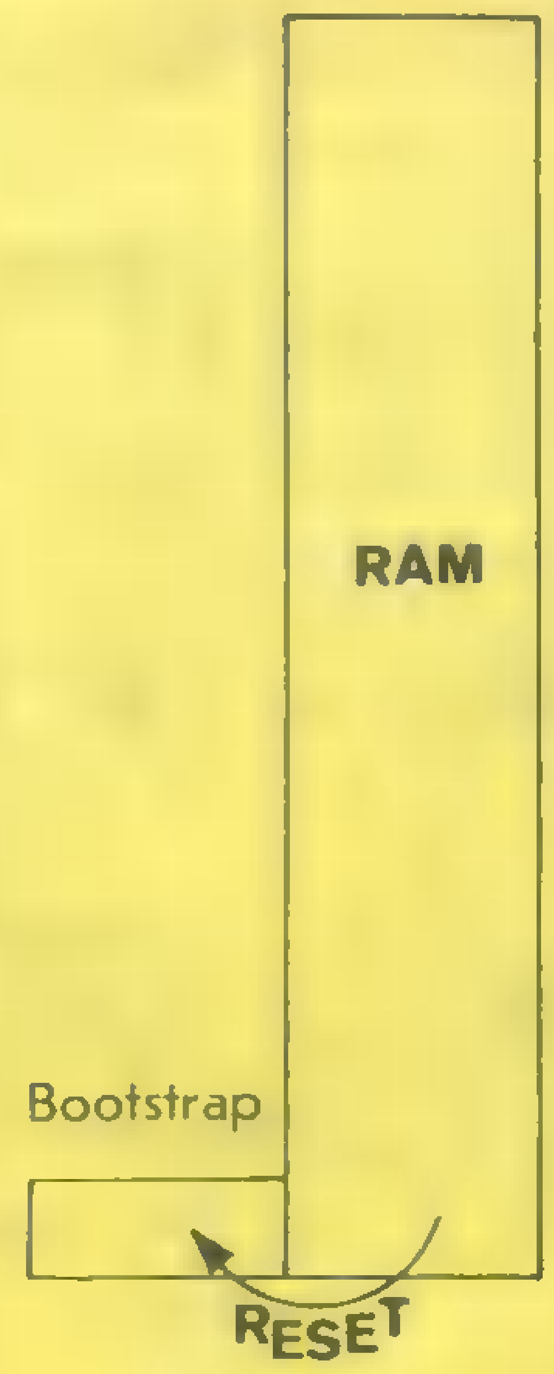
6. Dostęp do przerwań maskowalnych i niemaskowalnych.

Przedyskutujmy teraz podłączenie pamięci RAM tak, aby obejmowała obszar od 0 do 3FFFH. W mikrokomputerze ZX Spectrum przewidziano możliwość wyłączenia pamięci ROM. Na złączu krawędziowym dostępny jest sygnał ROMCS. Po podaniu na tą linię logicznej jedynki pamięć ROM przestaje być widoczna dla procesora. Ten sam sygnał może załączyć dodatkową pamięć RAM zgłaszającą się pod tym adresem. Przedstawiono to 3 (zakładamy tu istnienie pamięci RAM 16KB oznaczonej w opisie karcianym symbolem „pik"). W ten prosty sposób, korzystając

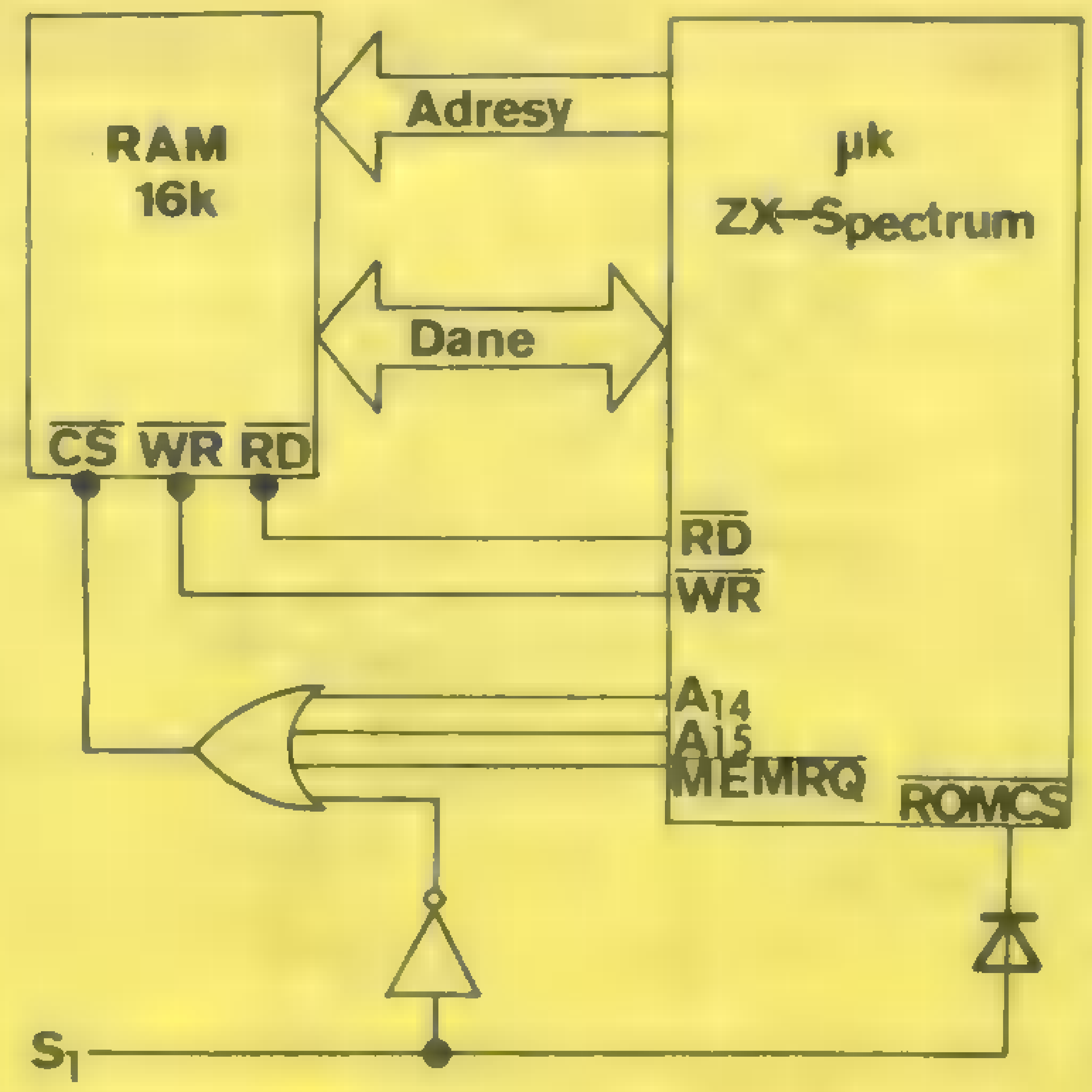




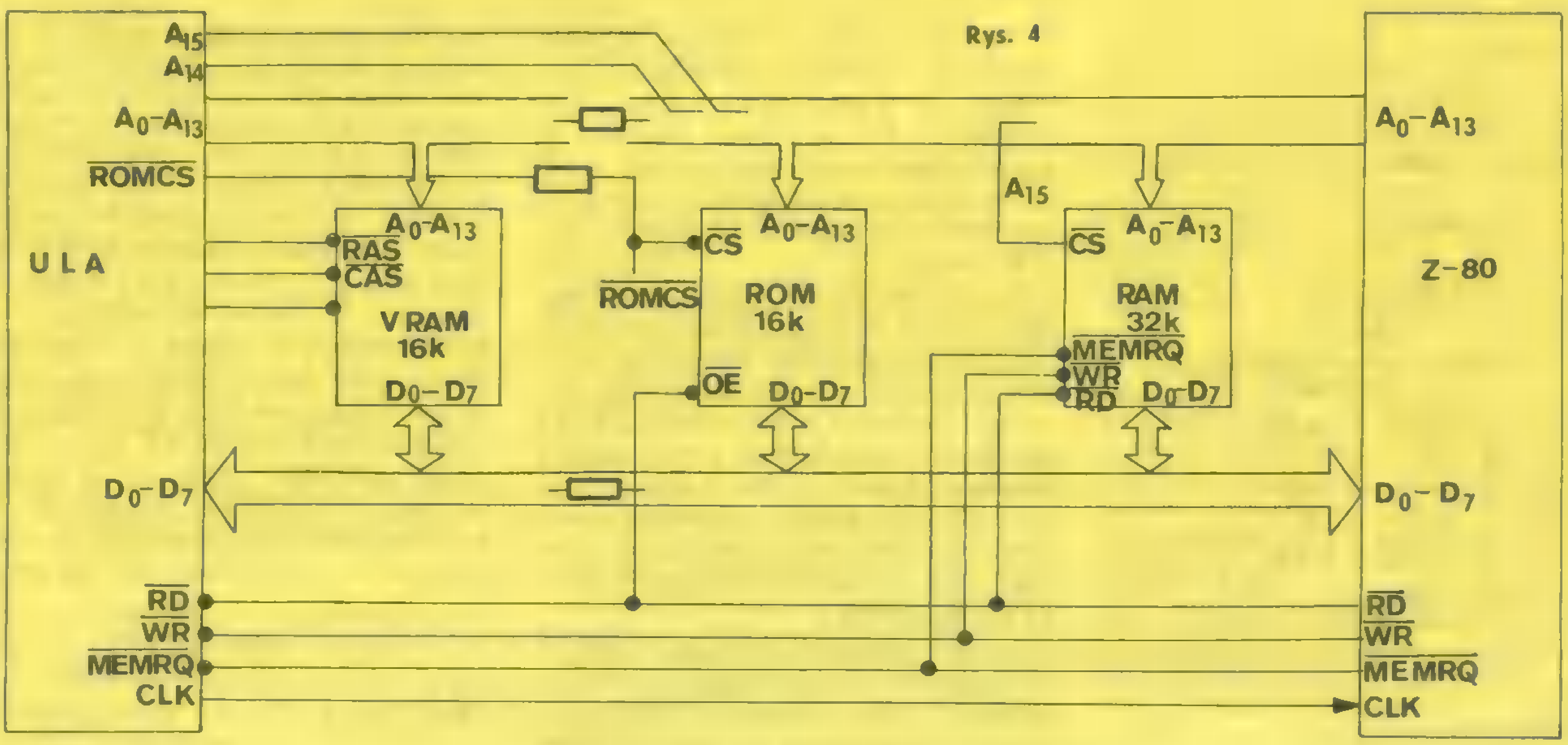
Rys. 1



Rys. 2



Rys. 3



Rys. 4

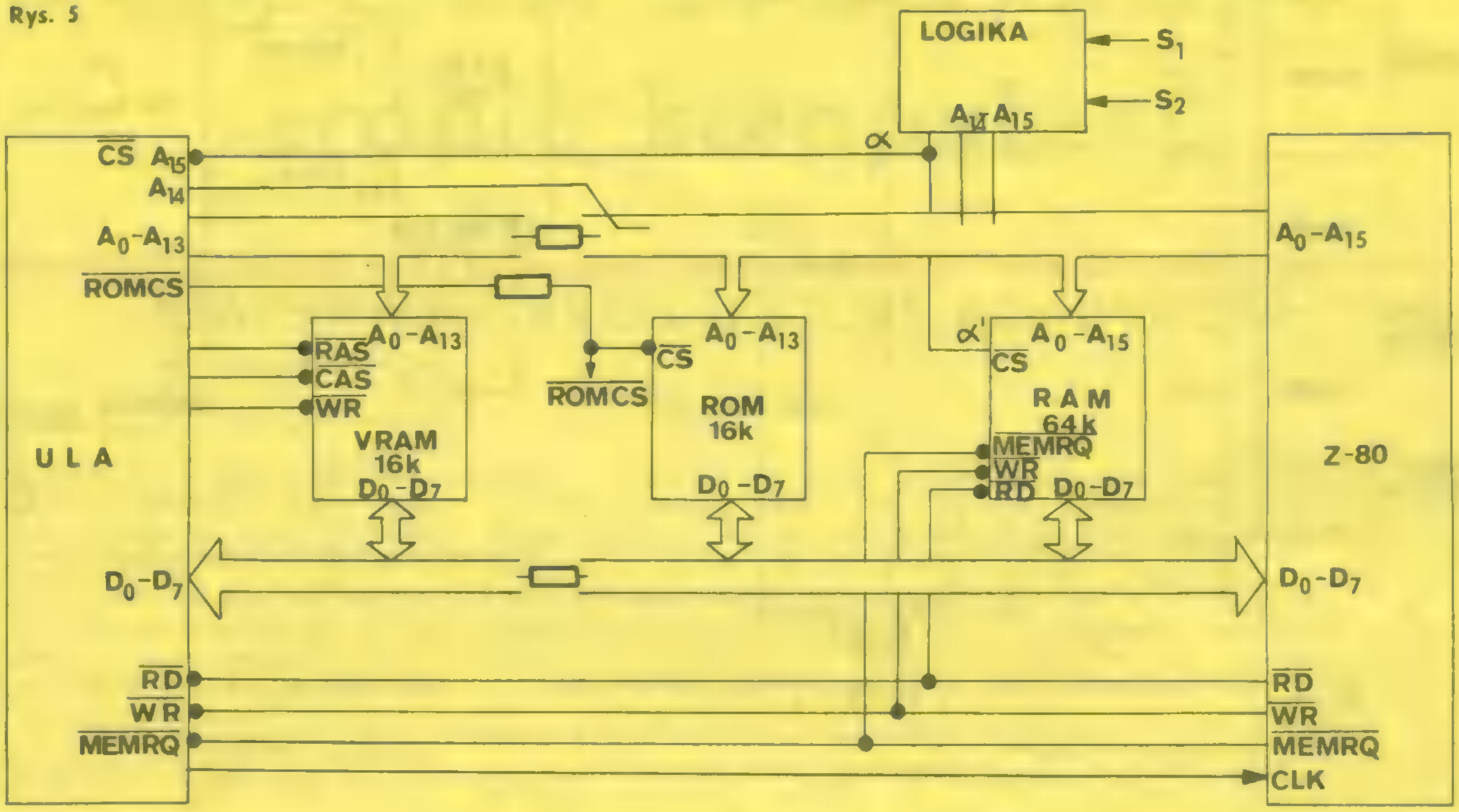
z sygnału ROMCS wyłączyliśmy ROM. Podobne rozwiązanie zastosowali konstruktorzy INTERFACE I. Jak się jednak dalej okaże, można rozwiązać ten problem zupełnie inaczej.

Spróbujmy przedyskutować teraz problem podłożenia pamięci RAM w miejsce drugiej ćwiartki przestrzeni adresowej mikroprocesora (oznaczonej „kier"). Problem nie jest łatwy i aby go rozwiązać, należy przypatrzeć się pracy mikroprocesora w II i III ćwiartce pamięci.

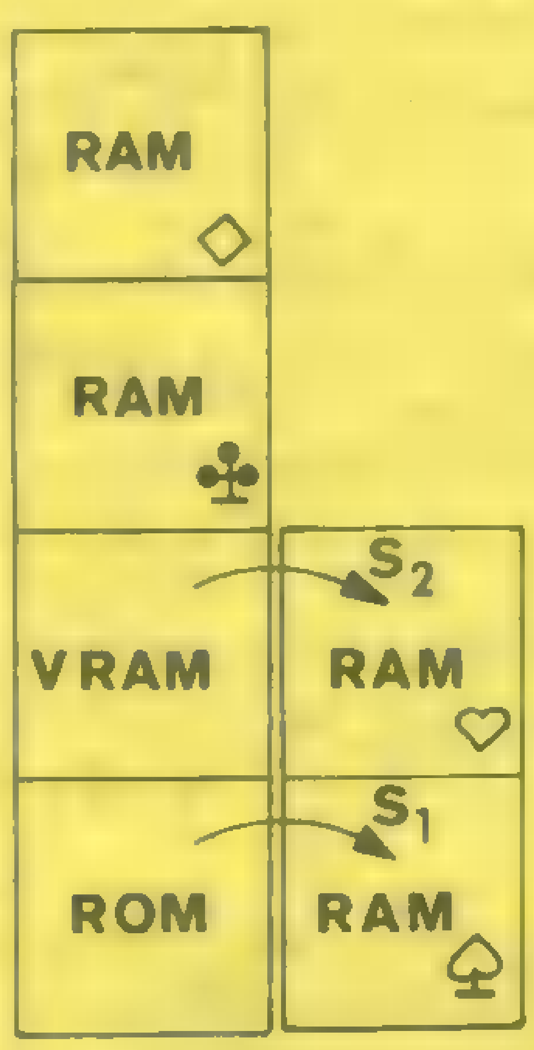
W III ćwiartce mikroprocesor adresuje pamięć RAM, a ta zgłasza się w cyklu RD (READ-odczyt) lub WR (WRITE-zapis). Jednocześnie w mikrokomputerze zachodzi drugi, całkowicie niezależny proces wyświetlania obrazu. Procesor graficzny (ULA) adresuje pamięć VRAM, a ta zgłasza się w cyklu RD. Możliwe jest więc zgłaszanie się w tej samej chwili pamięci VRAM i procesora lub pamięci RAM. Problem konfliktów na magistrali rozwiązany jest za pomocą rezystorów, które rozdzielają szynę danych na szynę procesora i szynę ULA (patrz rys. 4). Rezystory te pozwalają na to, aby w mikrokomputerze ZX Spectrum przebiegały jednocześnie i niezależnie dwa procesy: proces tworzenia obrazu przez ULA i praca mikroprocesora operującego na obszarach pamięci RAM „trefl" i „karo".

Przypatrzmy się cyklowi dostępu do pamięci w II ćwiartce przestrzeni adresowej. Procesor w stanie T1 wystawia adres, który informuje ULA o żądaniu dostępu do pamięci VRAM. W chwili, gdy ULA adresuje pamięć procesor nie może otrzymać do niej dostępu. Zostaje więc zatrzymany w stanie T2 do chwili, gdy ULA zwolni swoją część magistrali. ULA wygeneruje wtedy sygnały sterujące pamięć VRAM (RAS, CAS, WR) dla adresów procesora. Dostępy procesora do VRAM w czasie wygaszania sygnału wizji odbywają się bezkonfliktowo.

Rys. 5



Rys. 6



Taka metoda synchronizacji procesów powoduje wydłużenie czasu wykonywania jednego z nich. W ten sposób można wytłumaczyć fakt, że programy w języku maszynowym umieszczone w przestrzeni VRAM pracują o około 20% wolniej.

Z rozważań tych wynika, że gdy mikroprocesor nie adresuje pamięci VRAM, to jego praca nie jest niczym zakłócana, a procesor graficzny (ULA) kontroluje pamięć VRAM jako niezależny proces. Podłączenie dodatkowej pamięci RAM „kier" jest możliwe wtedy, gdy pod adresami II ćwiartki przestrzeni adresowej nie będzie zgłaszała się pamięć VRAM. Można to uzyskać traktując nóżkę A15 układu ULA jako sygnał CS. Sygnał ten zablokuje ULA w taki sposób, że nie będzie on w stanie generować sygnałów sterujących dla VRAM i ROM (RAS, CAS, WR i ROMCS). Ten sam sygnał powinien otwierać dodatkowy blok pamięci RAM 32KB (bloki „pik" i „kier"). Sygnał ten będziemy oznaczać w opisie literą α (rys. 5, 6, tablica).

Przypatrzmy się tablicy funkcji α. Z pracy w trybie (3) wynika, że procesor adresuje bezpośrednio 64K pamięci RAM, a więc swą pełną przestrzeń adresową. Wymagany jest więc blok pamięci 64 KB podłączony bezpośrednio pod magistralę procesora. Posiadacze ZX Spectrum 16K mogą rozwiązać problem wkładając w podstawki pamięci 64K. Posiadacze wersji 48K mogą natomiast wymienić elementy 32 KB na 64 KB.

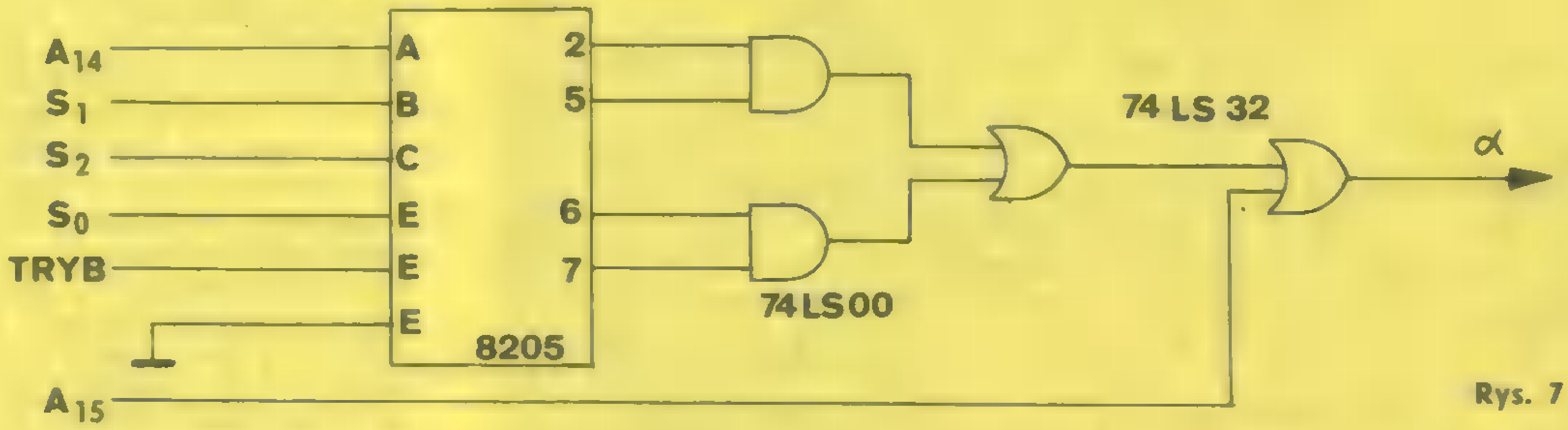
Przypatrzmy się układowi sterowania pamięci RAM (rys. 8). Widać wyraźnie, że każde wystąpienie sygnału MEMRO powoduje generację sygnału RAS, co umożliwia zapis, odczyt, a także odświeżenie tej pamięci. Cykle zapisu i odczytu są rozpoznawane na bramce * pod warunkiem, że pamięć ta jest wybrana (A15 = 1 na bramce **). Powstaje sygnał MUX przełączający multipleksery, a potem sygnał CAS odblokowujący bufory pamięci. Odświeżanie powoduje tylko generację sygnału RAS, ponieważ w takim cyklu pamięciowym nie są generowane sygnały RD i WR. Jak widać, sygnał A15 wykorzystany jest w ZX Spectrum jako CS bloku 32 KB. W wersji 48K producent umieszcza uszkodzone pamięci 64 KB, wybierając je tak, aby komplet 8 kostek miał te same dobre połówki (32 KB). Wyboru tych połówek można dokonać za pomocą odpowiednich zwor (w wersji 2 możliwy jest wybór tylko jednej z dwóch połówek). Wkładając w miejsce pamięci 32 KB komplet dobrych kostek 64 KB (np. 4164) wykorzystamy tylko 32 KB. Ale, gdy wprowadzimy na zworę (nóżka 11 układu IC 26, czyli wejście 3A multipleksera) sygnał A15, to procesor adresować będzie pełne 64 KB przy czym sygnał α podłączony do bramki ** (patrz rys. 8) uaktywniać będzie cały blok. Jeżeli podłączymy sygnał α z wyjściem układu generującego sygnał α to uzyskamy interesującą nas mapę pamięci (rys. 6).

Posiadacze kostek 64 KB wykorzystują wejście A15 bloku pamięci 64 KB do przełączania dwóch bloków pamięci 32 KB od adresu 8000H. Rozwiązanie takie przedstawiono na rys. 9. Często jest ono propo-

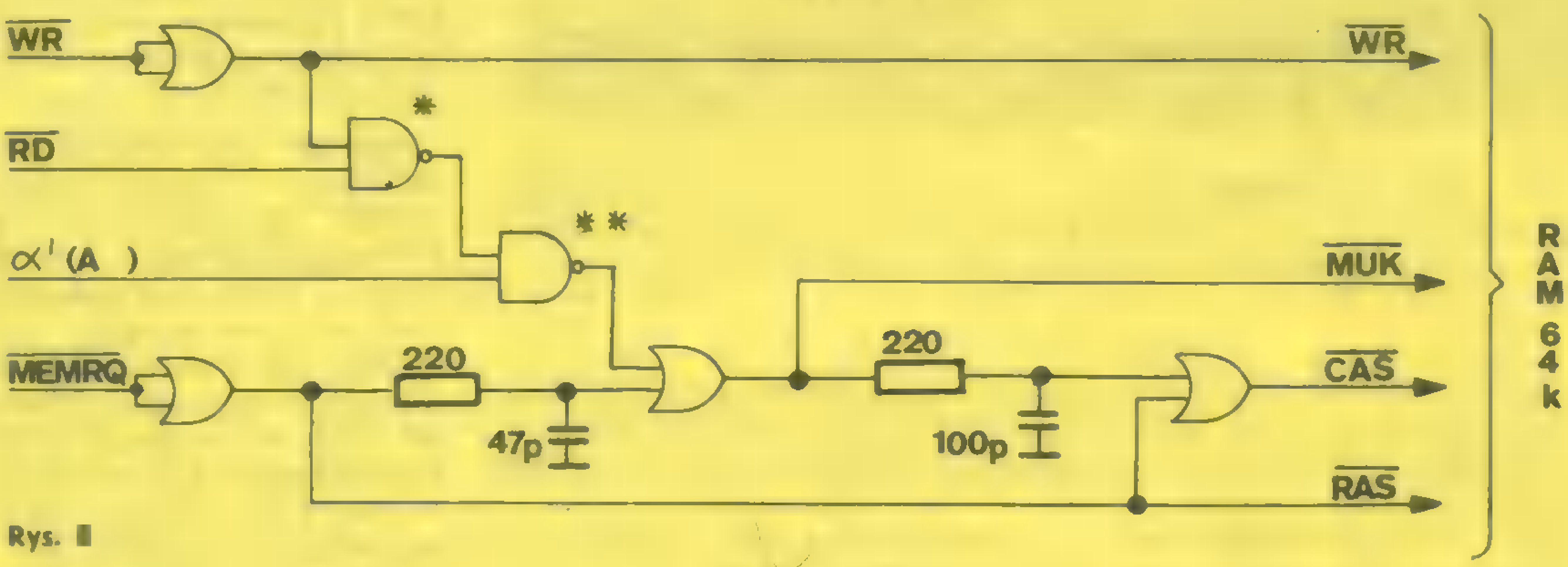
Tablica 1

α

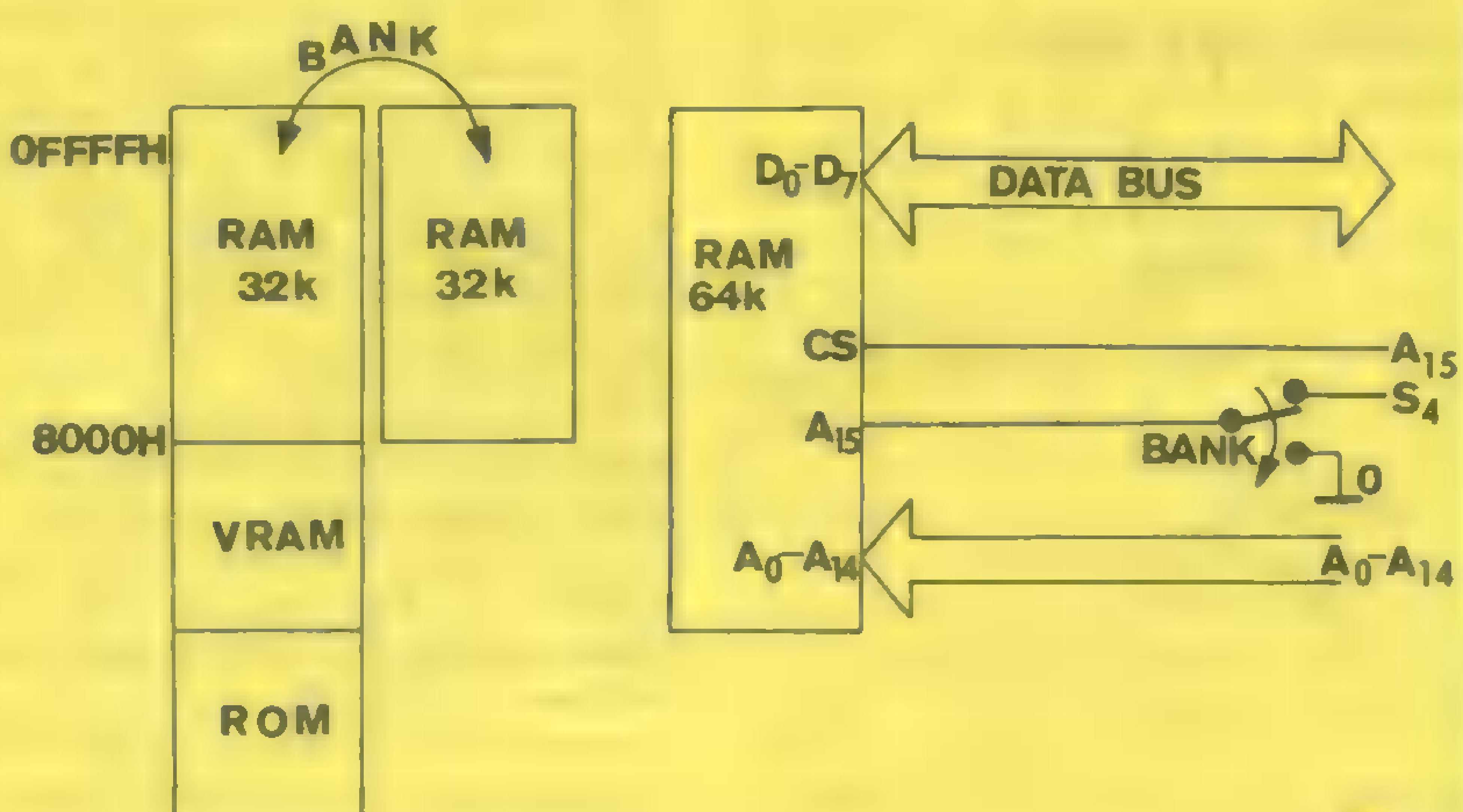
S_2 S_1 \ A_{15} A_{14}	0 0	0 1	1 1	1 0	
0 0	0	0	1	1	(1)
0 1	1	0	1	1	(2)
1 1	1	1	1	1	(3)
1 0	0	1	1	1	(4)
	I ćw	II ćw	IV ćw	III ćw	



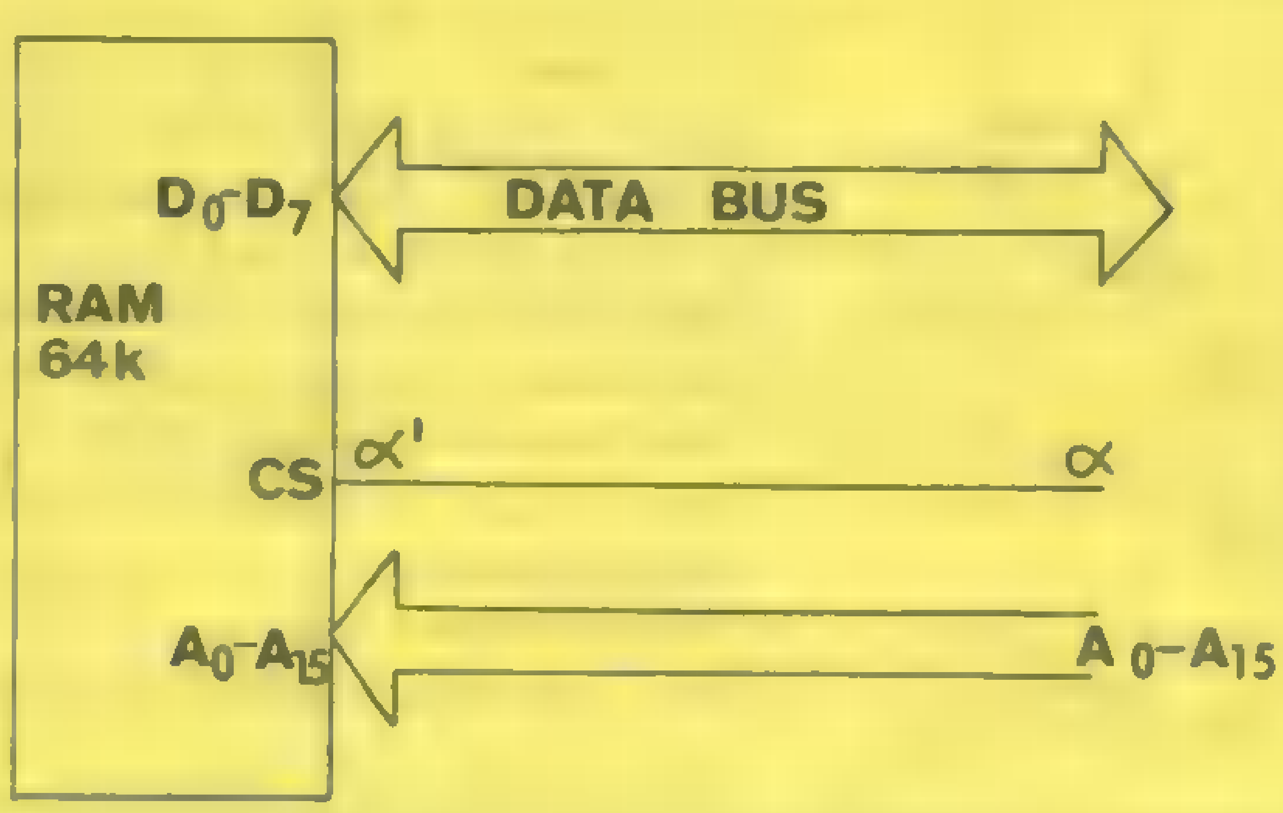
Rys. 7



Rys. 8



Rys. 9



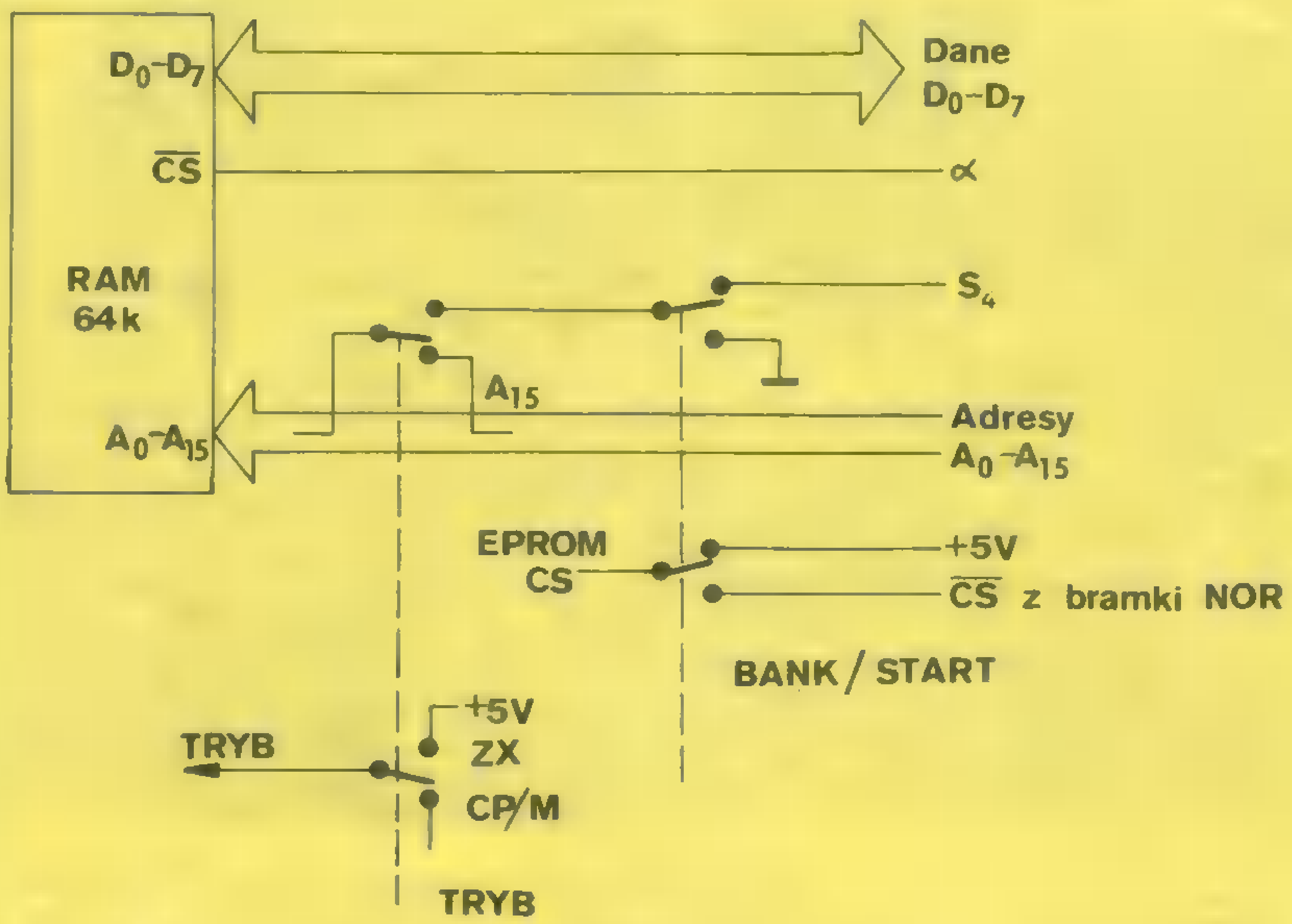
Rys. 10

nowane w zachodnich czasopismach pomimo braku większych zalet. Można jednak, bez dodatkowych kosztów, dobudować je do naszego rozwiązania, tak aby jeszcze bardziej zwiększyć elastyczność systemu.

Przypatrzmy się przełączaniu stron pamięci. Sygnał BANK wprowadzany jest na wejście A15 bloku pamięci 64 KB i wybiera jedną z dwóch połówek. Sygnał ten może być generowany z przełącznika, jak na rys. 9 lub z dodatkowego portu tak, aby przełączenia tego mógł dokonać mikroprocesor.

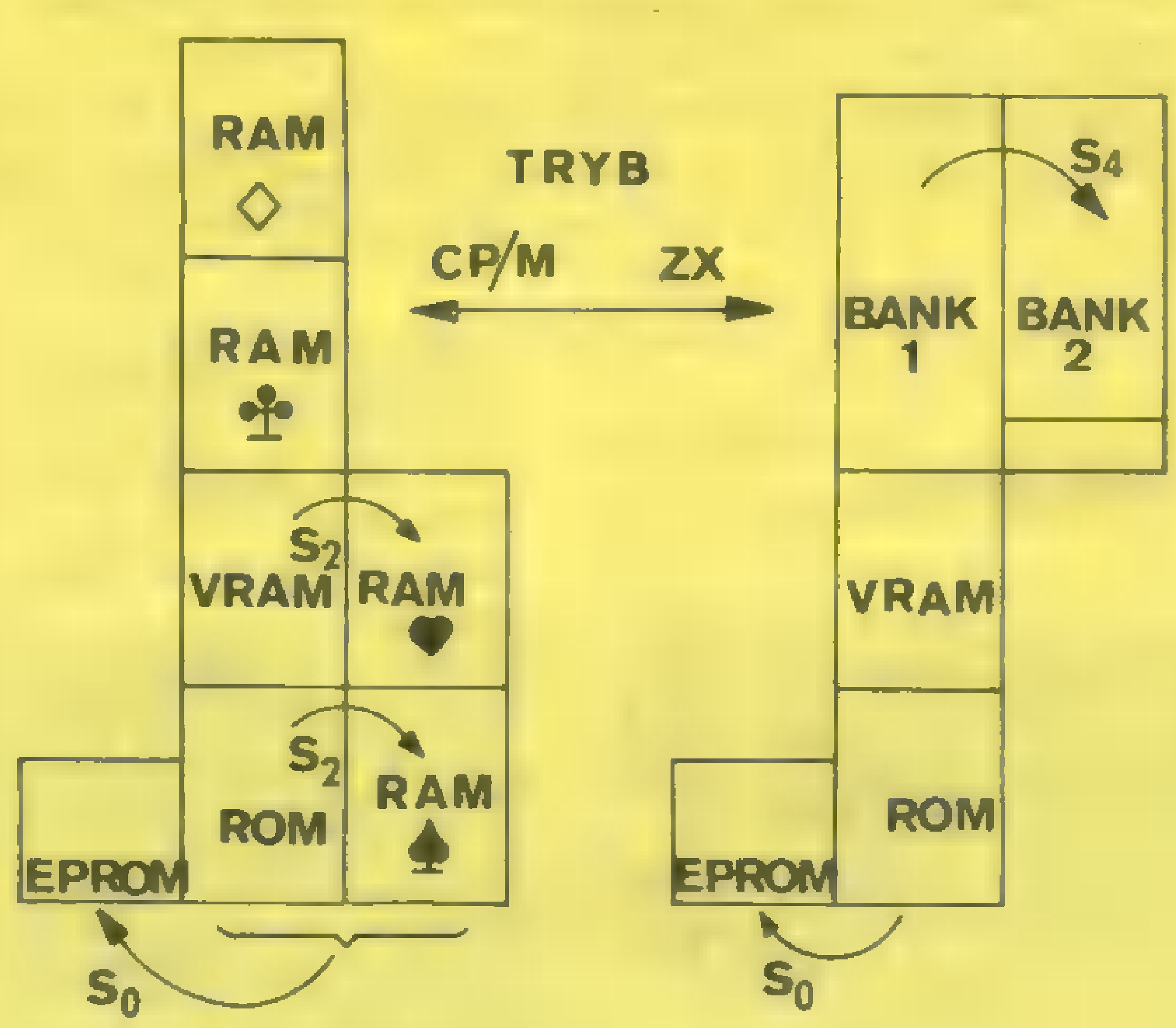
W poprzednim rozwiązaniu układ sterowania pamięcią 64 KB wyglądał inaczej (por. rys. 10 i rys. 11). Dla połączenia tych dwóch rozwiązań należy zastosować przełącznik zmieniający tryb pracy komputera (nazwane tu CP M< — >ZX). Przełączanie wejścia A15 pamięci odbywa się ręcznie. Jednocześnie na wejście CS pamięci podawany jest sygnał α, który w przypadku zablokowania dekodera 8205 sygnałem TRYB jest równy A15. Sygnał BANK można podłączyć tak, aby mógł być przełączany zarówno przez mikroprocesor, jak i ręcznie. Przełączanie ręczne jest możliwe, gdy S4 jest w stanie wysokim (patrz rys. 9), a przełączanie programowe, gdy przełącznik jest w odpowiedniej pozycji. Uzyskaliśmy w ten sposób mapę pamięci przedstawioną na rys. 12.

Jeżeli po załączeniu komputera sygnały sterujące S1 i S2 będą wyzerowane, to otrzymamy konfigurację normalnego ZX Spectrum, tzn. od 0 do 3FFFH ROM,

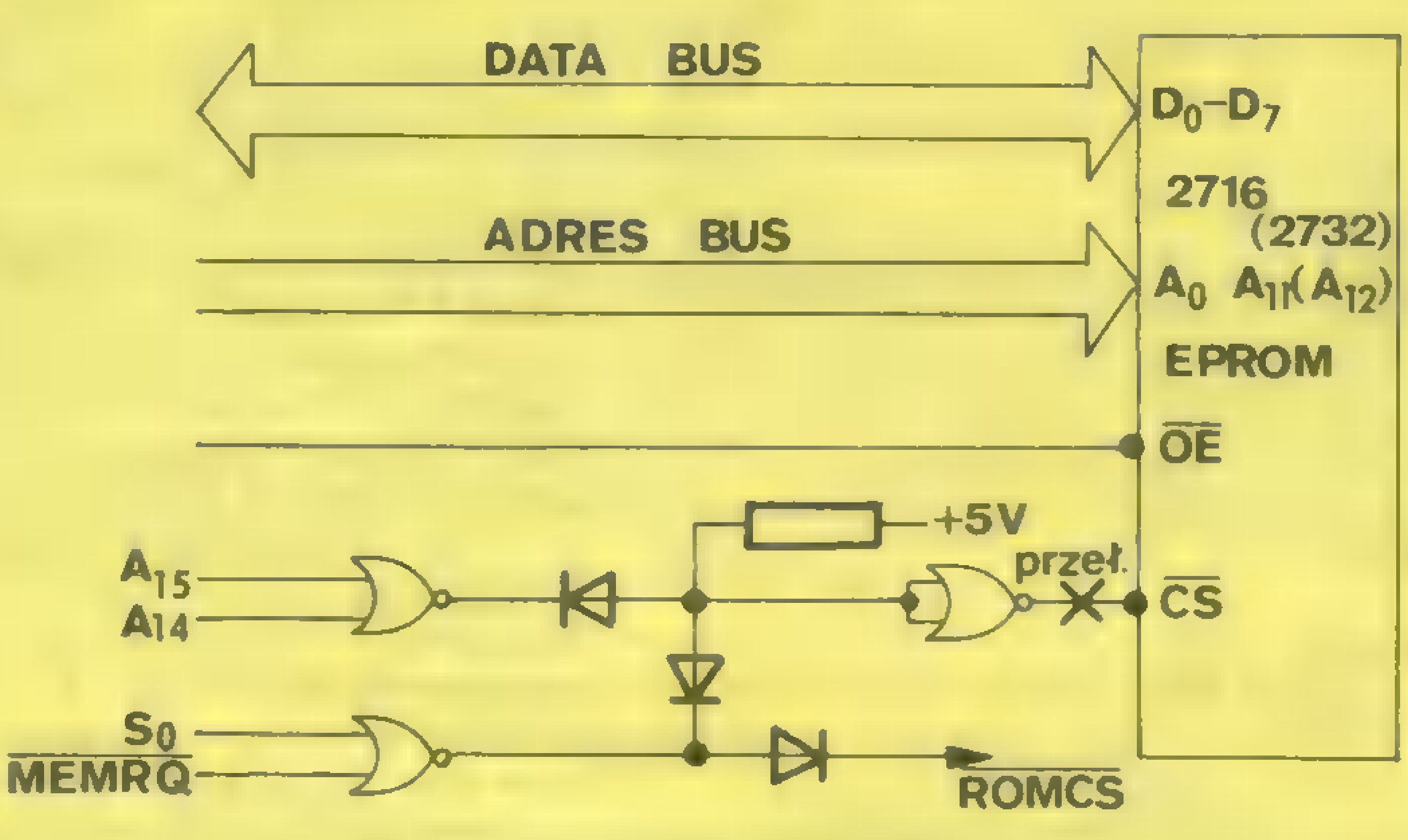


Rys. 11

Rys. 12



Rys. 13



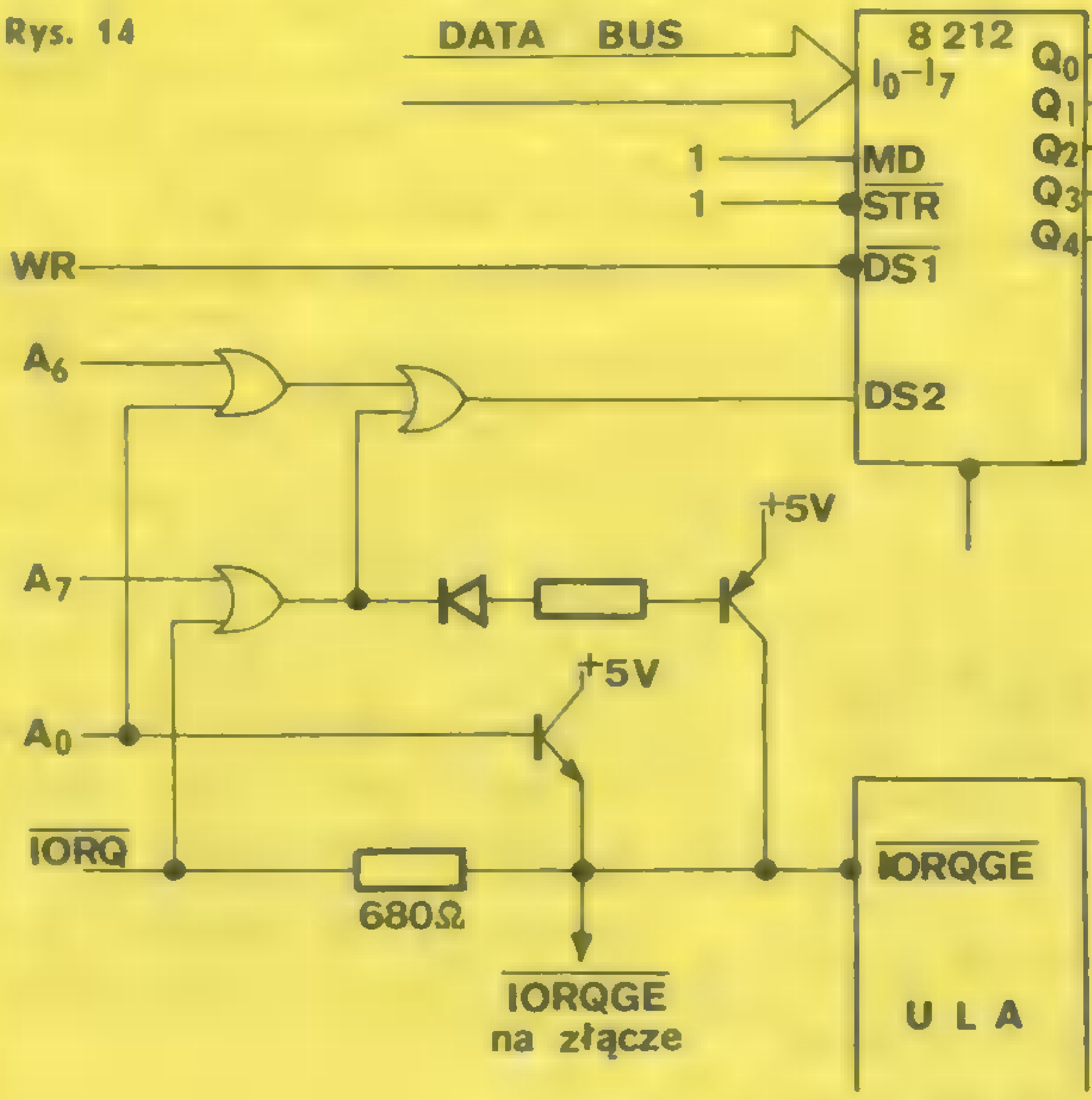
w obszarze 4000H—7FFFH VRAM, a powyżej tego RAM „karo" i „trefl". Jeżeli dobudujemy dodatkową pamięć ROM (EPROM) zawierającą program startowy, komputer zgłaszając się, zapyta o tryb pracy. Będzie wtedy można wybrać pracę z normalnym interpreterem BASIC-a z ROM-u i przełączaniem górnej połowy pamięci (sygnał BANKSWITCH i przełącznik), pracę ze zmienionym interpreterem przepisanym do pamięci RAM „pik" lub pracę pod kontrolą innych systemów (np. wczytaniu systemu CP/M z dysku lub monitora z mikrokomputera COBRA). Do sterowania dodatkową pamięcią z programem startowym można wykorzystać sygnał SO, który będzie aktywny w stanie L, tak aby RESET komputera uaktywniał pamięć EPROM. Sygnał ten, połączony z sygnałem MEMRQ procesora blokuje pamięć ROM. Przykładowe rozwiązanie sterowania dodatkową pamięcią ROM przedstawione jest na rys. 13.

Niepełne adresowanie pozwala na zastosowanie tu różnych typów pamięci EPROM. Np. układ 2708 o pojemności 1 K zgłaszać się będzie 16 razy, a 27128 tylko raz. Użycie tego pierwszego nie jest jednak wskazane ze względu na kłopotliwe zasilanie, zaś drugiego, z uwagi na zbyt dużą pojemność. Program startowy swobodnie mieści się w pamięci 4 KB (2732). Zastosowana w tym rozwiązaniu logika diodowa, a konkretnie bramka AND działa poprawnie w myśl zasady: „gdy się nie ma co się lubi..." Należy jednak pamiętać, że bramki TTL-LS mają stosunkowo mały prąd wypływający z wejść w stanie L, dlatego użycie opornika w bramce diodowej jest niezbędne. Przy sygnałach sterujących S0 = 0 i S1 = 1 na magistrali danych powstanie konflikt polegający na zgłaszaniu się pamięci EPROM i pamięci RAM „pik". Aby temu zapobiec, sygnał S0 blokuje układ generacji sygnału α, a konkretnie dekoder 8205 (porównaj rys. 7).

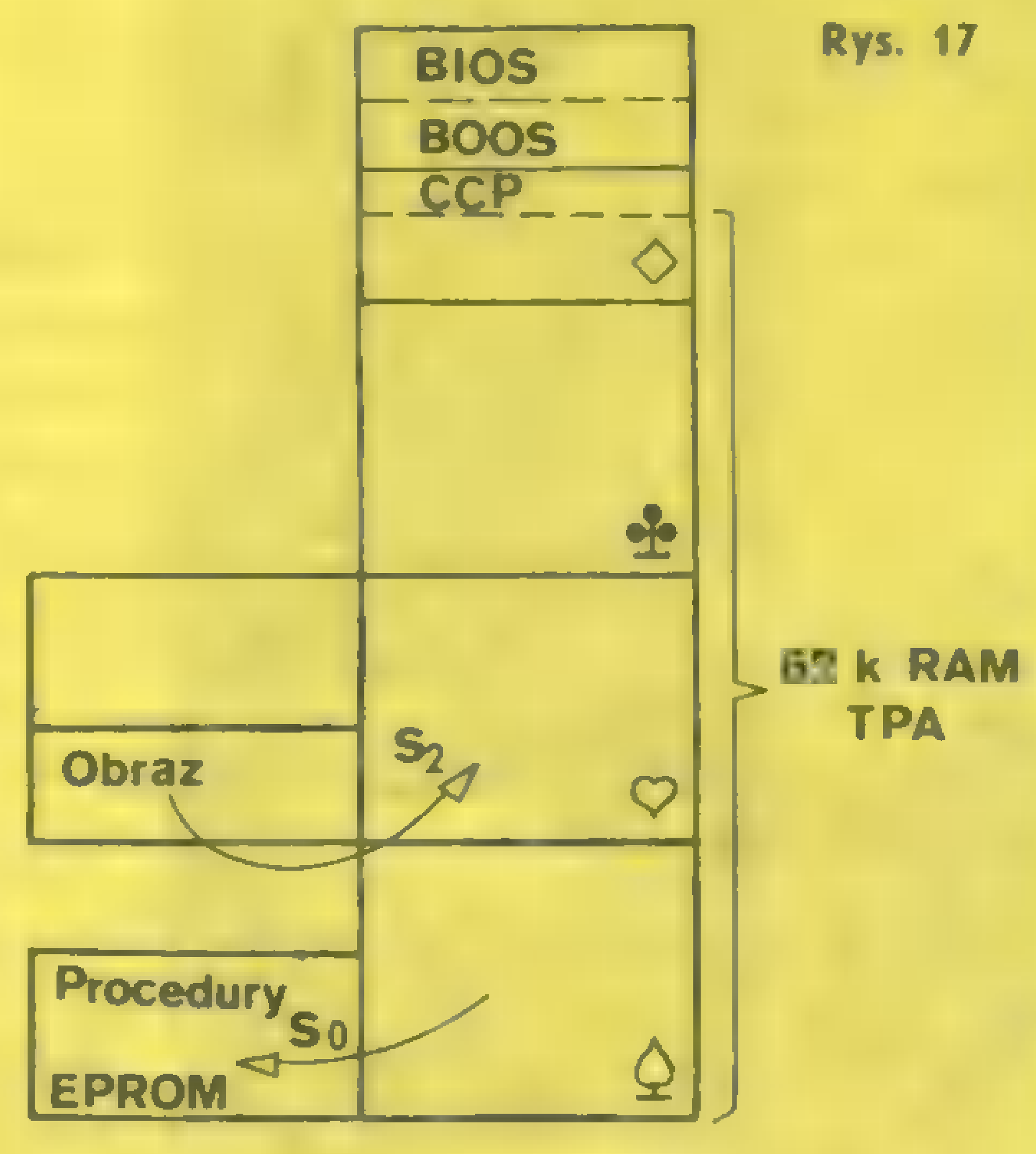
Przed montowaniem układu należy przedyskutować jeszcze pochodzenie sygnałów S0-S4. Sygnały te mają być tworzone programowo przez procesor, dlatego należy podłączyć do komputera dodatkowy port wyjściowy i wykorzystać 5 z jego 8 bitów. Problem powstaje przy analizie przestrzeni adresowej I/0 — większa jej część jest już wykorzystana. Wszystkie adresy parzyste wybierają port ULA (A0 = 0). Stan L na A2 wybiera ZX Printer i inne drukarki, a bity A3 i A4 sterują pracą INTERFACE 1. Ponieważ ten dodatkowy port zainstalowany będzie we wnętrzu komputera, wskazane jest, aby nie ograniczał on i tak już małej przestrzeni dostępnej na zewnątrz (tzn. bity A1, A5, A6 i A7). Interpreter języka BASIC odwołuje się do portu ULA wystawiając adres FEH. Można zatem wybierać ten port w bardziej precyzyjny sposób ograniczając przestrzeń adresową zajmowaną przez ULA. Na rys. 14 przedstawiony jest sposób podłączenia portu 8212.

W oryginalnym mikrokomputerze ZX Spectrum port ULA zgłasza się w cyklu

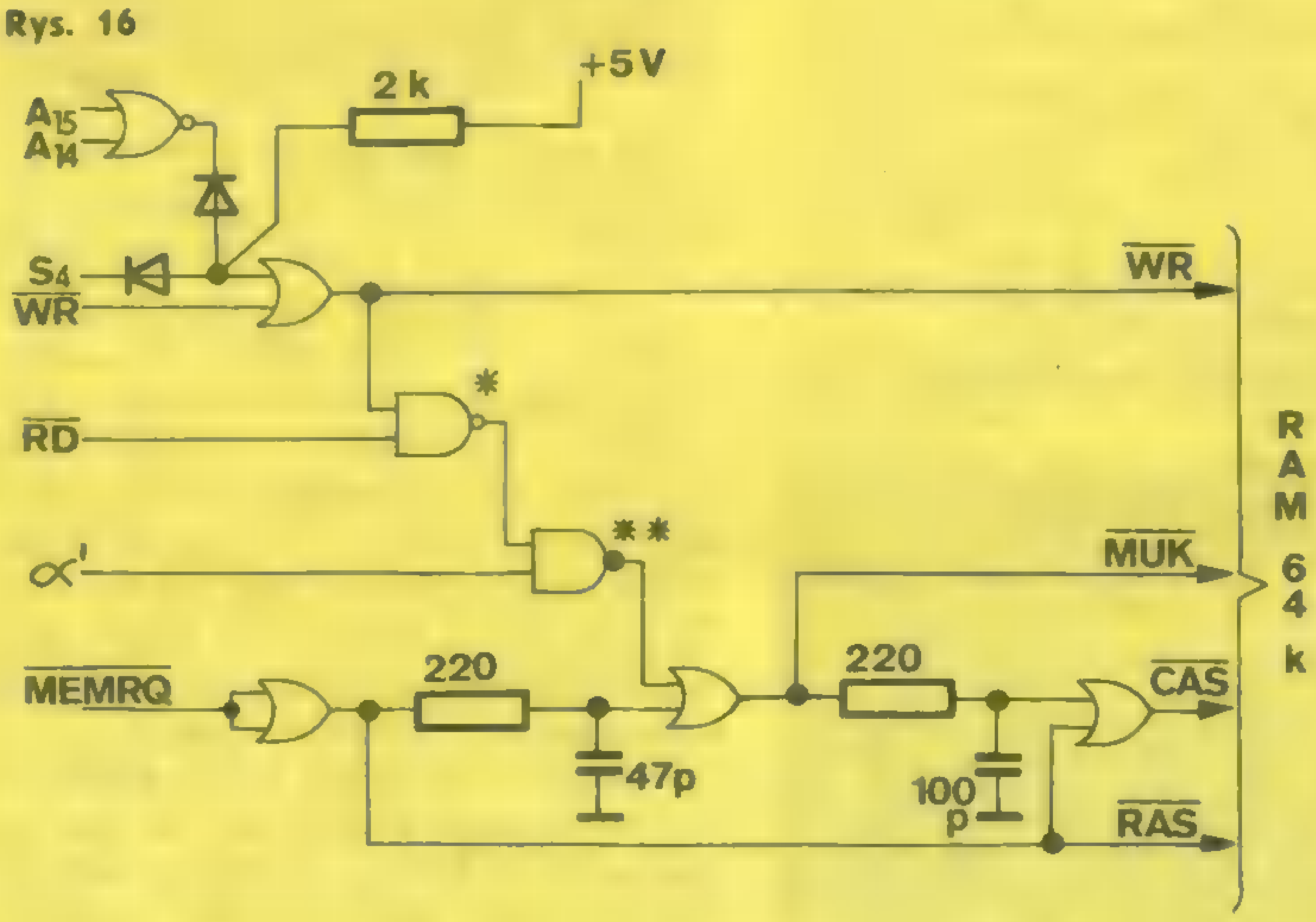
Rys. 14



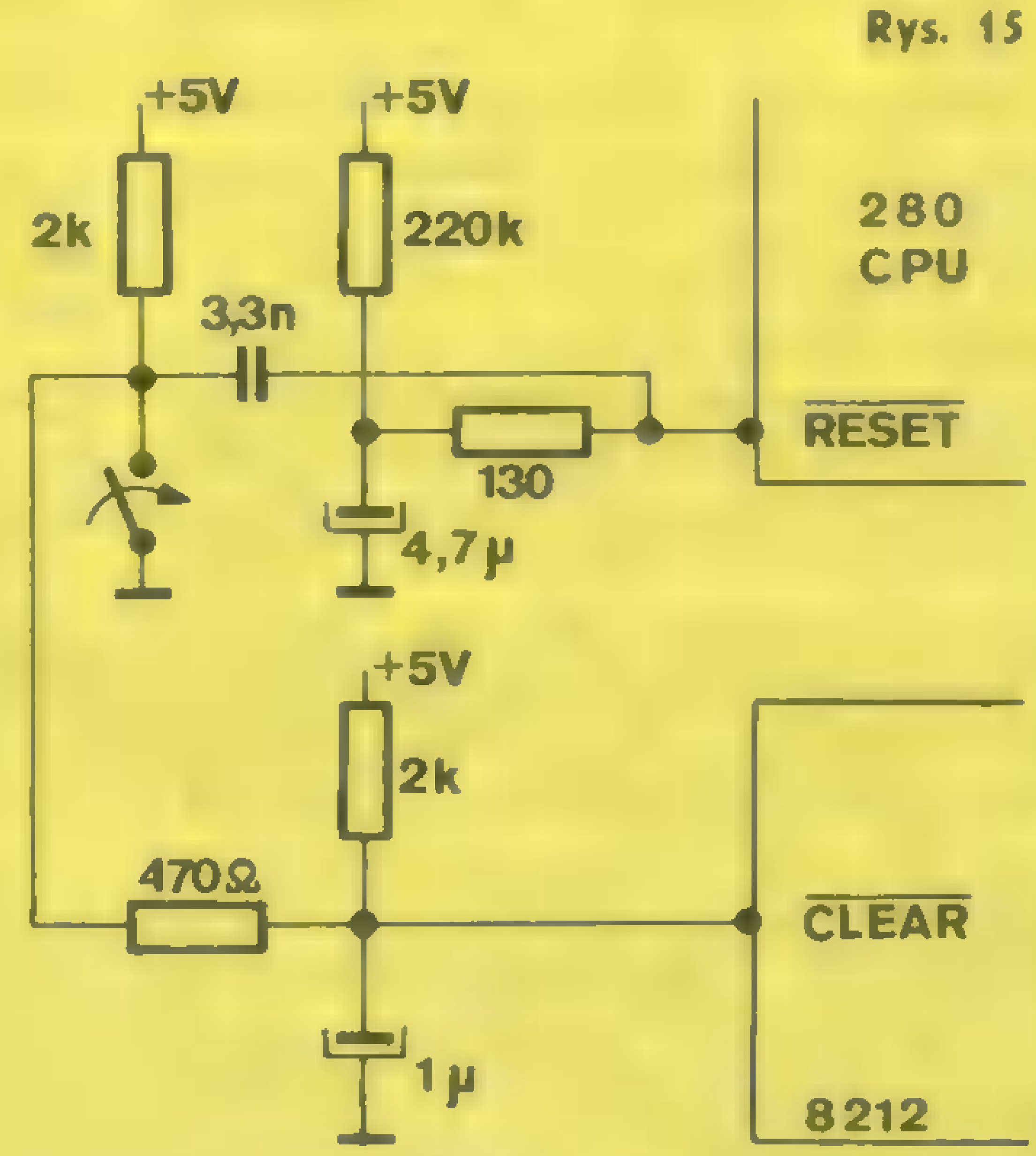
Rys. 17



Rys. 16



Rys. 15



I, 0 gdy A0 = 0; w przerobionym zaś, gdy A0 = 0 i A7 = 1. Sposób wymuszenia stanu H przez tranzystor nie jest zbyt elegancki, ale jak widać na schemacie, producent mikrokomputera także stosuje takie sposoby. Tak więc ULA zgłasza się pod adresami 1XXXXXXOB, a nasz port pod 00XXXXXOB; adresować go można np. 3EH lub 62D. W układzie bardzo ważna jest dioda obniżająca napięcie załączające tranzystor — w praktyce najlepiej zastosować tu diodę LED lub BAP 813, gdyż spadek napięcia w kierunku przewodzenia na tych elementach wynosi ok. 2V. Ograniczenie przestrzeni adresowej ULA nie zmniejsza możliwości mikrokomputera, ponieważ wszystkie odwołania pod ten port w ROM-ie i w innych programach wystawiają adres FEH. Dodatkowy port nie będzie więc w konflikcie z żadnym urządzeniem zewnętrznym.

Wyjścia portu tworzą sygnały:

S0 — Bootstrap
S1 — RAM „pik"/ROM
S2 — RAM „kier"/VRAM
S3 — Write-disable
S4 — Bank-switch

Pozostałe nie są wykorzystane i można tam podłączyć np. dodatkowe diody świecące umieszczone na płycie czołowej komputera lub drugi głośniczek.

Po załączeniu komputera generowany jest sygnał RESET zerujący licznik programu mikroprocesora. W rozbudowanym komputerze sygnał ten powinien zerować sygnały sterujące S0—S4. Port 8212 posiada wejście CLEAR, ale podłączenie go pod RESET procesora nie da zadowalających efektów. Na rys. 15 przedstawiono rozwiązanie pozwalające zerować port i procesor przy załączeniu komputera oraz zerować procesor przyciskiem. Rezystor 470 ohm spowoduje zerowanie 8212 przy dłuższym załączeniu przycisku. Analizując ten układ należy pamiętać, że komputer może pracować poprawnie tylko wtedy, gdy sygnał CLEAR z 8212 ustąpi.

Po przepisaniu interpretera BASIC-a do pamięci RAM „pik" stwierdzono, że pierwsze pięć komórek zmieniło swe wartości. Okazuje się, że interpreter zapisany w ROM-ie niszczy sam siebie — twórcy oprogramowania ułatwili sobie zadanie zakładając, że ich dzieło zapisane będzie na stałe w pamięci ROM. I tak na przykład przejście przez procedurę SKIP-CONSTANS (33F7H) niszczy 5 pierwszych komórek pamięci RAM „pik", a niektóre funkcje edytora niszczą matryce znaków graficznych. Tak więc, aby nie przerabiać za bardzo interpretera BASIC-a, do układu dorobiono sprzętowe zabezpieczenie przed zapisem w pamięci RAM „pik". Podobne



zabezpieczenie stosuje się w dużych komputerach (np. w systemie operacyjnym GEORGE-3 na ODRZE-1305) dla uniemożliwienia zniszczenia systemu operacyjnego. Na rys. 16 przedstawiony jest układ blokujący sygnał WR dla bloku pamięci 64KB, gdy sygnał S3 jest aktywny (stan H) i jednocześnie procesor adresuje pierwsze 16 KB (tzn. pamięć „kier"). Zablokowanie sygnału WR i brak sygnału RD spowoduje, że cykl takiego dostępu zostanie odczytany przez pamięć jako cykl odświeżania. Podobnie, jak przy sterowaniu pamięci EPROM, zastosowana tu logika diodowa działa od ponad roku poprawnie.

Program inicjujący komputer w trybie ZX umieszczony jest, jak już wspomniano w pamięci EPROM (listing). Usuwa on błąd w obsłudze przerwania nie maskowanego, zmienia *copyright* i dołącza do interpretera następujące instrukcje:

Pusch	—	przeładowuje RAM „trefl" do RAM „kier"
Pop	—	przeładowuje RAM „kier" do RAM „trefl"
Call W	—	skacze do podprogramu napisanego w języku wewnętrznym rozpoczynającego się od adresu W
Monitor	—	w tej wersji nic nie robi (RET)

Instrukcje te włącza się przez nie wykorzystane w Spectrum (bez INTERFACE 1) słowa FORMAT, CAT, MOVE i ERASE. Zmiany w interpreterze można rozwijać wprowadzając instrukcje z BETA-BASICA, pod warunkiem, że zmieszczą się w 700 bajtach między kalkulatorem, a tablicą znaków. Można też całkowicie zmienić interpreter BASIC-a na np. popularny BASIC-Microsoft zaadaptowany z MERITUM. Ale nie w rozszerzeniu interpretera tkwi siła takiego systemu. Zobaczmy jak wygląda system CP/M przystosowany do takiego komputera.

Prawie cały system jest umieszczony w pamięci RAM „karo"; prawie cały tzn. BDOS i CCP, bo z BIOS-a tylko krótkie programy przełączające (patrz rys. 17).

Np. działanie procedury Co (Console Output) wygląda tak:
ustaw stos lokalny (w BIOS-ie);
włącz VRAM;
włącz EPROM;
skocz do procedury CO umieszczonej w EPROM-ie;
napisz znak na ekranie (matryca znaków w pamięci EPROM lub VRAM);
powróć do BIOS-a;
wyłącz VRAM, włącz RAM A;
odtwórz wskaźnik stosu;

Takie podejście pozwoliło skrócić BIOS-a i przesunąć do góry cały system. Zwiększa to przestrzeń programów TPA do 62 KB. Przy pracy pod CP/M układ Bootstrap może wczytać z dysku system operacyjny.

Przy pracy w trybie CP/M przełącznik BANK jest nie wykorzystany, można więc za jego pomocą wybierać rodzaj startu komputera. Przy pracy w trybie CP/M komputer może zgłaszać się pamięcią ROM lub pamięcią EPROM. Wyboru tego można dokonać odłączając sygnał S0 od układu sterowania BOOTSTRAP. Nie zostanie wtedy wygenerowany sygnał otwierający EPROM i sygnał zamykający ROM. W trybie ZX położenie przełącznika przy starcie ma podobny wpływ na pracę komputera; start z ROM-u zmusi nas do wykonania instrukcji OUT 62, BIN 10001 tak, aby można było korzystać z przełącznika BANK bez załączania BOOTSTRAP, natomiast start z EPROM-u wykona sprawdzenie trybu i sam ustali sygnały portu.

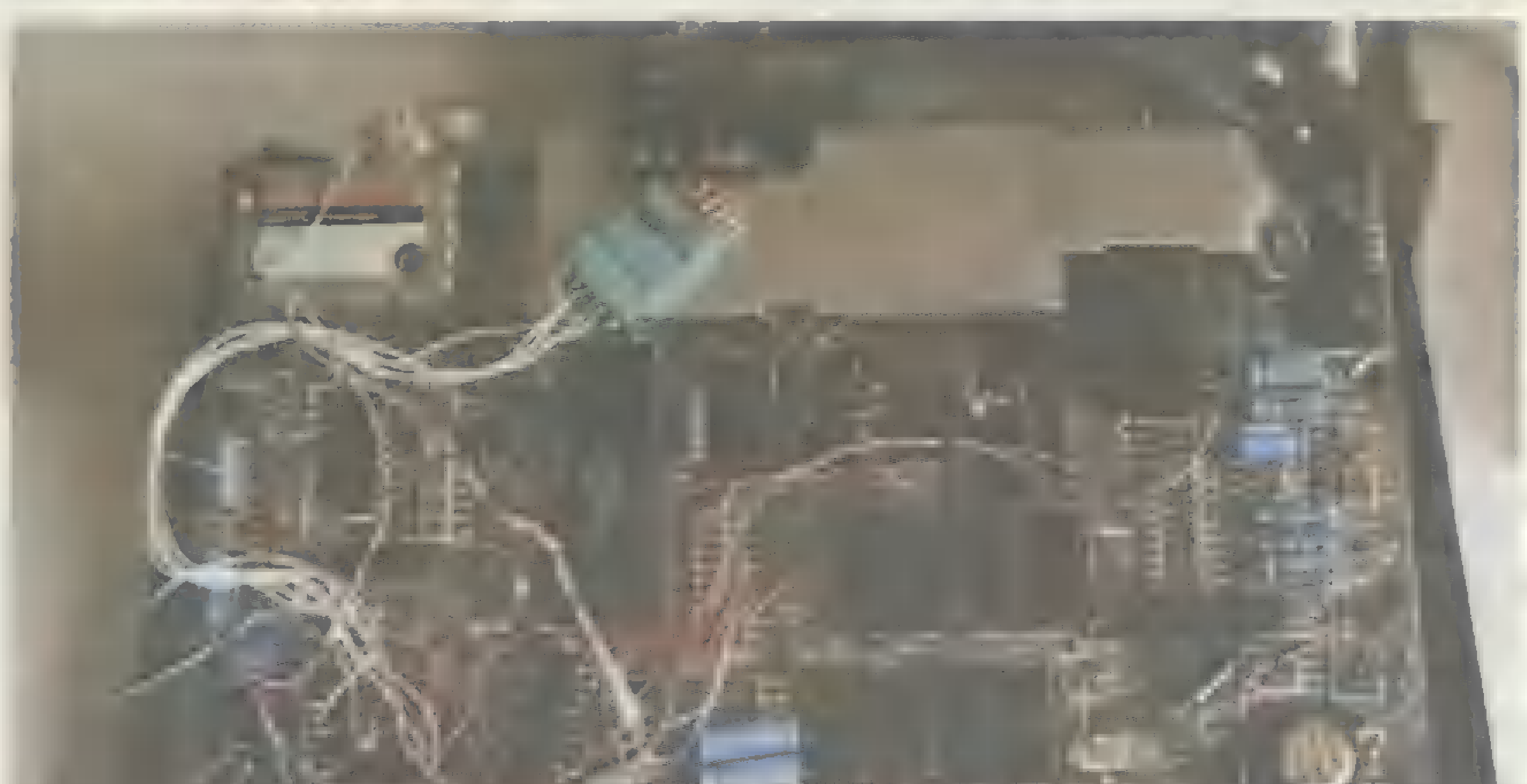
A teraz kilka słów o montażu układu. Założeniem przeróbek było zmieszczenie wszystkiego we wnętrzu normalnej obudowy. Posiadacze wersji drugiej komputera mogą umieścić płytkę uniwersalną w miejscu, gdzie inne wersje mają radiator. Posiadacze wersji trzeciej i czwartej mogą rozwiązać to montażem „na pająka".

Do układu potrzebne są następujące elementy:

74LS02 — jest polski!
74LS32 — trudny do zdobycia
8205 — jest polski odpowiednik UCY74S405
8212 — lub UCY74S412
EPROM 2716 lub 2732, 2764

W miejsce dwóch bramek NAND potrzebnych do tworzenia sygnału α można wykorzystać połowę 74LS00 istniejącego w komputerze. Można, ale tylko w wersji 2 i 3. W wersji 4 bramki te zostały wykorzystane, jako opóźnienie sygnału MUX sterującego multiplekserami pamięci VRAM. Posiadacze tej wersji mogą dołożyć jeden układ scalony lub, co jest prostsze, przerobić swoją wersję na wersję 3, tzn. zastąpić dwie bramki rezystorem 130 ohm, który wraz z pojemnością wejść stanowi takie samo opóźnienie. Teraz przecinając kilka ścieżek uwolnimy te dwie bramki. Kłopotliwe połączenie pod układem IS 24 wykonane od strony elementów można usunąć nawiercając wiertłem 0,8 mm otwór koło nóżki 12 układu IS 24.

Montaż można rozpocząć od uruchomienia portu wyjściowego. Jedynym miejscem dla tej dużej kostki jest nałożenie jej na kostki 4116 pod kątem, tak aby końcówki zasilania można było przylutować do szyn doprowadzających 5V i 0V do pamięci. Pozostałe nożki można powyginać do góry. Szyna danych procesora, do której należy podłączyć ten port, dostępna jest na opornikach R1-R8. Dla tranzystora zamykającego ULA najlepszym miejscem będzie mikroprocesor. Można położyć go płasko, a jego emiter podłączyć do nóżki 11 IC 2 czyli do Vcc. Pozostałe kostki można nałożyć na układy scalone, wluto-



wane już w układzie, wyginając przy tym ich nóżki do góry. Nóżki zasilające mogą służyć, jako konstrukcja wsporcza. Wszystkie połączenia należy wykonywać przewodami wlutowując je w przeloty istniejące na płytce. Ścieżkę A15 trzeba odciąć od ULA i RAM przy nóżce 20 procesora oraz koło gniazda magnetofonowego (MIC). Następnie należy go zmostkować tak, aby sygnał A15 procesora dostępny był na złączu krawędziowym.

W wersjach 3 i 4 zainstalowanie pamięci EPROM jest dość kłopotliwe. Optymalnym dla niej miejscem jest pamięć ROM. Podstawkę pamięci EPROM można nalutować na ROM mimo tego, że ma ona tylko 24 nóżki. Wynika to stąd, że rozmieszczenie interesujących nas linii adresowych, a także linii danych i masy jest identyczne w obu elementach. Zgadza się ułożenie końcówek A0-A11, D0-D7 i GND, pozostałe tzn. OE, CS i Vcc należy odgiąć i doprowadzić odpowiednie sygnały przewodami. Kłopotliwe może być jedynie zdobycie cienkiej podstawki pod EPROM, tak aby kostka ta zmieściła się pod radiatorem.

Należy zwrócić uwagę na zasilanie mikrokomputera. Po podłączeniu układów prąd pobierany z zasilacza wzrasta o około 200 mA. Należy więc tak ustawić napięcie za transformatorem, aby na wejściu komputera nie było więcej niż 9,5 V. Fabryczne zasilacze często dają pod obciążeniem napięcie około 12—14 V, co często jest przyczyną uszkodzeń komputera. Wyższe napięcie zasilania może spowodować „zatkanie" przetwornicy (chwilowe zatrzymanie pracy) i brak napięcia — 5V, co niszczy pamięci 4116.

OD AUTORA

Pisząc powyższy artykuł miałem na celu zachęcenie posiadaczy mikrokomputera ZX Spectrum do samodzielnego przerabiania własnego sprzętu. Pomysł przeróbek powstał w sierpniu 1985 roku, a prototyp w listopadzie. Tak długi czas i liczne dyskusje nad układem pozwoliły uniknąć wielu kłopotów z uruchomieniem. Opis ten, w pełni oddając tok mojego rozumowania, może innym ułatwić dokonanie identycznych lub tylko częściowych przeróbek swoich mikrokomputerów. Tych, którzy zdecydują się na wszystkie proponowane przeze mnie przeróbki zapraszam do wspólnego oprogramowania. Dostrzegłem cztery dziedziny, którymi warto się zainteresować:

1) Zmiany interpretera ZX Spectrum — to dla tych, którzy lubią rozszerzania BASIC-a i dobrze znają ROM;

2) Przenoszenie oprogramowania z innych komputerów (MERITUM, ARMSTAD, LASER, COBRA);

3) Adaptacja systemu CP/M 2.2, CP/M+ oraz ISIS ze stacją dysków 5 ¹/₄ cala;

4) Stworzenie nowego systemu operacyjnego dopasowanego do sprzętu i służącego do badań nad systemami operacyjnymi.

Osobiście interesują mnie badania systemowe i zamierzam zająć się dwoma ostatnimi kierunkami pozostawiając pozostałe innym. W celu koordynacji działań wszystkich przerabiających proponuję utworzenie sekcji przerabiaczy przy klubie mikrokomputerowym ENIAC. Od zainteresowanych oczekuję zgłoszeń pod adresem:

Klub Mikrokomputerowy
ENIAC
sekcja sprzętowa
DS „ONDRASZEK"
Gliwice,
ul. Kujawska 2, pok. 427
lub pod adresem redakcji.

Przyjemnie jest mi w tym miejscu złożyć podziękowania wszystkim moim kolegom, którzy swoimi radami przyczynili się do powstania zarówno prototypu, jak i tego artykułu.

```
  10 Switc:         EQU  %00111110
  20 ERRNR:         EQU  23610
  30 NMIRE:         EQU  #5CB0
  40 CHARS:         EQU  23606
  50 NEW:           EQU  #11B7
  60 CR:            EQU  13
  70
  80                ORG  0
  90
 100 Reset:
 110
 120                ORG  23755+5  ;w REM
 130
 140 Boost:         DI
 150                LD   HL,Start
 160                LD   DE,#8000
 170                LD   BC,DlugB
 180                LDIR
 190                JP   #8000
 200
 210 Start:         DI
 220                LD   A,%0001  ;ROM on
 230                OUT  (Switc),A
 240                LD   HL,0
 250                LD   DE,#4000
 260                LD   B,D
 270                LD   C,E
 280                LDIR
 290                LD   A,%0011  ;ROM off
 300                OUT  (Switc),A
 310                DEC  DE
 320                DEC  HL
 330                EX   DE,HL
 340                LD   BC,#4000
 350                LDDR
 360
 370 ; petla zmian interpretera
 380
 390                LD   HL,TabZm-Start+#8000
 400 Loop:          LD   C,(HL)
 410                INC  HL
 420                LD   B,(HL)
 430                LD   A,B
 440                CP   C
 450                JR   Z,Exit
 460                INC  HL
 470                LD   E,(HL)
 480                INC  HL
 490                LD   D,(HL)
 500                INC  HL
 510                LDIR
 520                JR   Loop
 530 Exit:          LD   A,%1011  ;blokada on
 540                OUT  (Switc),A
 550                JP   Reset
 560
 570 ;************************************
 580 ;
 590 ; Tablica zmian interpretera
 600 ;
 610 ; organizacja
 620 ;      defw dlugosc blokul
 630 ;      defw adres blokul
 640 ;      defb bajty do wymiany
 650 ;      ....
 660 ;      defb bajty do wymiany
 670 ;      defw 0  ;straznik
 680 ;
 690 ;************************************
 700
 710
 720 TabZm:
 730                DEFW D1Zm1
 740                DEFW #1539
 750 x1:            EQU  #1539-$
 760                DEFM " Przerwanie"
 770                DEFB CR
 780                DEFM "   niemaskowane"
 790                DEFB "!"+128
 800 D1Zm1:         EQU  $+x1-#1539
 810
 820                DEFW D1Zm2
 830                DEFW #1295
 840 x2:            EQU  #1295-$
 850                CALL InMNI
 860                XOR  A
 870 D1Zm2:         EQU  $+x2-#1295
 880
 890                DEFW D1Zm3
 900                DEFW #06D
 910 x3:            EQU  #06D-$
 920                DEFB #28
 930 D1Zm3:         EQU  $+x3-#06D
 940
 950                DEFW D1Zm4
 960                DEFW #04AA
 970 x4:            EQU  #04AA-$
 980 InMNI:         EQU  $+x4
 990                LD   HL,ObNMI
1000                LD   (NMIRE),HL
1010                LD   HL,NEW
1020                LD   A,%00011  ;WRenable
1030                OUT  (Switc),A
1040                LD   (5+1),HL
1050                LD   A,%01011  ;WRdisable
1060                OUT  (Switc),A
1070                LD   DE,NowyC
1080                RET
1090 D1Zm4:         EQU  $+x4-#04AA
1100
1110                DEFW D1Zm5
1120                DEFW #386E
1130 x5:            EQU  #386E-$
1140 ObNMI:         EQU  $+x5
1150                LD   IY,ERRNR
1160                LD   (IY),28
1170                CALL #16C5
1180                EI
1190                JP   #1303
1200 ObCal:         EQU  $+x5
1210                CALL #1E99
1220                LD   H,B
1230                LD   L,C
1240                CALL #162C
1250                LD   IY,ERRNR
1260                EI
1270                RET
1280 ObPus:         EQU  $+x5
1290                LD   HL,#8000
1300                LD   DE,#4000
1310 obpop:         LD   BC,#4000
1320                DI
1330                LD   A,%1111
1340                OUT  (Switc),A
1350                LDIR
1360                LD   A,%1011
1370                OUT  (Switc),A
1380                EI
1390                RET
1400 ObPop:         EQU  $+x5
1410                LD   HL,#4000
1420                LD   DE,#8000
1430                JR   obpop
1440 NowyC:         EQU  $+x5
1450                DEFB " "+128
1460                DEFM "# 1985 ENIAC'owe"
1470                DEFM " Spectrum"
1480                DEFB CR
1490                DEFM "  Wersja 1.1 "
1500                DEFM "  1986-01-0"
1510                DEFB "5"+128
1520 ObMon:         EQU  $+x5
1530                RET
1540 D1Zm5:         EQU  $+x5-#386E
1550
1560                DEFW D1Zm6
1570                DEFW #11F
1580 x6:            EQU  #11F-$
1590                DEFB "C","a","l","l"+128
1600                DEFM "Monito"
1610                DEFB "r"+128
1620                DEFB "P","u","s","h"+128
1630                DEFB "P","o","p"+128
1640 D1Zm6:         EQU  $+x6-#11F
1650
1660                DEFW 3
1670                DEFW #1B14
1680                DEFB 3
1690                DEFW ObCal
1700
1710                DEFW 3
1720                DEFW #1B06
1730                DEFB 0
1740                DEFW ObMon
1750
1760                DEFW 3
1770                DEFW #1B0A
1780                DEFB 0
1790                DEFW ObPus
1800
1810                DEFW 3
1820                DEFW #1B10
1830                DEFB 0
1840                DEFW ObPop
1850
1860                DEFW 0
1870 DlugB:         EQU  $-Start
```

PRZERABIAMY VELĘ 203 NA MONITOR KOMPUTEROWY

Pół roku temu opisaliśmy, jak wykonać samodzielnie monitor komputerowy na bazie podzespołów NEPTUNA 150. Tym razem opiszemy bardzo prostą przeróbkę VELI 203, umożliwiającą dołączenie do niej praktycznie każdego komputera wyposażonego w standardowe wyjście VIDEO, czyli np. popularnego ATARI 800 XL.

Schemat ideowy układu dopasowującego jest przedstawiony na rysunku. Elementy dodatkowe wyróżnione są kolorową linią (po lewej stronie, na dole) — jak widać, jest ich rzeczywiście niewiele (trzy rezystory, jeden tranzystor, dwa kondensatory i jedna dioda). Wejście sygnału VIDEO (wyprowadzenie nr 1 gniazda diodowego) jest połączone z wejściem wtórnika na tranzystorze BC148 przez kondensator oddzielający składową stałą (220 nF). Kondensator ten pełni jeszcze jedną funkcję: współpracuje z diodą BAP795 tworząc prosty układ odtwarzania składowej stałej. Odpowiedni poziom napięcia stałego podanego na dalsze stopnie wzmacniacza wizji telewizora jest konieczny między innymi dla poprawnej pracy separatora impulsów synchronizacji — przy niezachowaniu odpowiedniej jego wartości następuje zrywanie synchronizacji, zależnie od treści obrazu.

Tor wizji odbiornika VELA nie posiada niestety odtwarzania składowej stałej — co gorsza ma on sprzężenie zmiennoprądowe. Objawia się to na ekranie tym, że średni poziom jasności pozostaje stały, niezależnie od treści obrazu (tła). Jest to bardzo niewygodne, powoduje zafałszowanie właściwego oddawania stopnia szarości obrazu. Można jednak, za pomocą bardzo prostego układu, poprawić stałość poziomu czerni ok. 2—3 razy w stosunku do oryginału — służy temu dioda BAVP19 i kondensator 10 nF w układzie sterowania jaskrawości. Działania ich są identyczne, jak układu odtwarzania składowej stałej w bazie tranzystora wtórnika. Elementy te montujemy w okolicy punktów 27 i 30 płytki radiowej, również bezpośrednio na druku.

Układ odtwarzania składowej stałej działa następująco: baza tranzystora wtórnika pobiera pewien niewielki prąd polaryzujący, w przypadku tranzystora npn wpływający do bazy. W czasie przekazywania treści obrazu prąd ten rozładowuje kondensator sprzęgający 220 nF — ze względu na krótki czas trwania jednej linii (64 mikrosekundy) rozładowanie to jest pomijalnie małe. W czasie trwania najbardziej ujemnej części sygnału VIDEO, czyli impulsu synchronizacji, następuje doładowanie kondensatora przez diodę BAP795 do poziomu napięcia wynikającego ze stopnia podziału dzielnika rezystorowego, a więc pewnego poziomu stałego, niezależnego od treści obrazu. Rezystory dzielnika tego zostały tak dobrane, aby napięcie na wyjściu układu dopasowującego było maksymalnie zbliżone do występującego na wyjściu detektora wizji.



Wyjście układu włączone jest na wejście pierwszego stopnia wzmacniacza bezpośrednio za detektorem wizji (baza T102). Przełącznik P (jedna jego połowa) realizuje odpowiednie przełączanie źródła sygnału wizji.

Sygnał fonii (dostępny na styku nr 3 gniazda diodowego) podawany jest bezpośrednio na wejście potencjometru regulacji siły dźwięku, przez drugą sekcję przełącznika P. Jest to najprostsze wyjście, choć amplituda sygnału jest trochę za mała w stosunku do wymaganej dla pełnego wysterowania wzmacniacza m.cz. Praktycznie jednak siła dźwięku jest zupełnie wystarczająca, ponadto w trosce o pozostałych domowników czasem lepiej nie denerwować ich zbyt silnymi odgłosami komputerowych sygnałów.

Układ najlepiej zmontować „w powietrzu" — ilość elementów jest na tyle mała, że można to wykonać bez przeszkód i obawy o zwarcia. Na płytce radiowej (lewa, patrząc od tyłu odbiornika po odkręceniu tylnej ścianki) należy przeciąć połączenie między dławikiem Dł106 i bazą T102 (rysunek) i odpowiednio, za pomocą niedługich przewodów (nie muszą być ekranowane), połączyć z przełącznikiem rodzaju pracy. Przełącznik ten powinien mieć dwa stabilne położenia — najlepiej zastosować pojedynczy isostat. Wszystkie elementy wtórnika oraz układu odtwarzania składowej stałej mieszczą się bez problemu tuż pod ekranem zespołu filtrów p.cz.

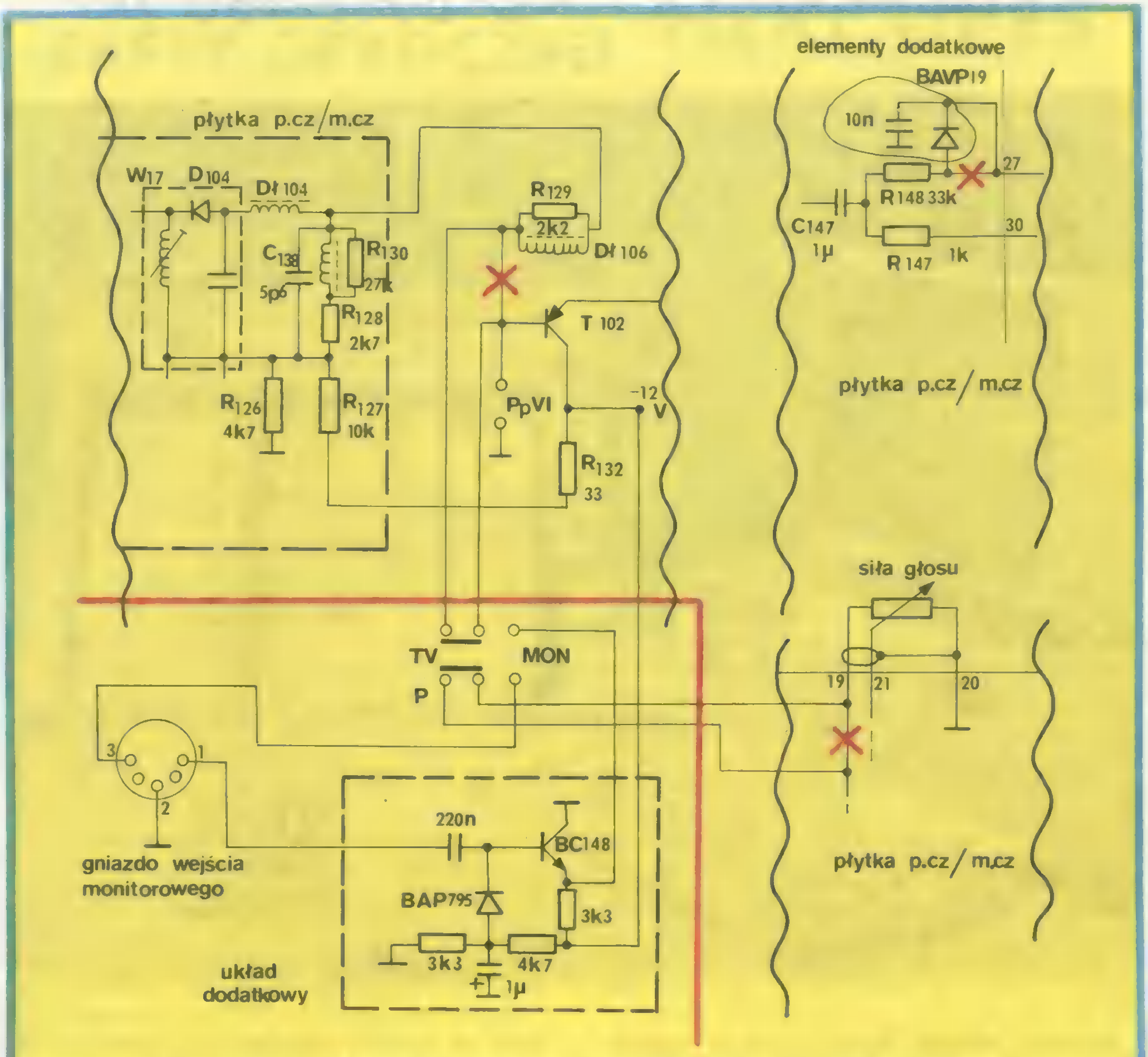
Dołączenie sygnału dźwięku jest jeszcze łatwiejsze. Wystarczy jedynie odlutować przewód ekranowy łączący punkt 19 płytki z potencjometrem i odpowiednio dolutować przewody do przełącznika i gniazdka wejścia AUDIO/VIDEO.

Opisany układ przystosowany jest do sygnału VIDEO o polaryzacji dodatniej i amplitudzie ok. 1,75 V. Na wyjściu popularnego Spectrum (wyjście Y — z układu ULA) dostępny jest sygnał o podobnej amplitudzie, lecz odwrotnej polaryzacji. Aby zatem możliwe było dołączenie do przerobionej VELI także Spectrum, konieczne jest wyposażenie go w inwerter o wzmocnieniu —1. Może to być prosty stopień z tranzystorem typu BC148 lub tp. z rezystorami o wartościach nie niżej 680 omów (warunek odpowiedniego pasma przenoszenia inwertera).

Grzegorz Załot

Wykaz elementów (razem z układem odtwarzania składowej stałej w obwodzie wzmacniacza wizji):

— tranzystor BC148 lub podobny
— dioda BAP795 i BAVP19 lub odpowiedniki
— kondensatory 10 nF/250 V, 220 nF/100 V, 1 F/25 V elektrolityczny
— rezystor (dowolna moc) 3k3 — 2 szt., 4 k7 — 1 szt.



SCHEMAT DOŁĄCZENIA WEJŚCIA MONITOROWEGO DO OTV VELA 203

Ciąg dalszy ze str. 6

pierwszego przeznaczony jest początkowy rozdział wprowadzający w zupełnie elementarne pojęcia związane z komputerami i informatyką (np. bit, bajt, mikroprocesor, urządzenie peryferyjne itp.). Kolejno autorzy omawiają zakres potencjalnych zastosowań C64 i opisują podstawowe zasady jego obsługi. Czwarty rozdział książki zainteresuje już zaawansowanego użytkownika. Dotyczy on bowiem rezydującej w C64 wersji języka BASIC V2. Opis jest bardzo drobiazgowy, a na szczególną uwagę zasługuje temat obsługi urządzeń zewnętrznych i komunikatów błędów.

Następny rozdział dotyczy zasad poprawnego programowania, a tym programowania strukturalnego. Później zamieszczony jest opis języka SIMON S BASIC. Każda instrukcja SIMON S-a opisana jest w punktach: działanie, składnia, opis parametrów i przykład. Rozdział kończy opis komunikatów błędów SIMON S BASIC-a, przykład programu w tym języku. Kolejno przedstawiono opis programowania w kodzie maszynowym. Opisano „serce C64 — mikroprocesor 6502, a także jego instrukcje asemblera wraz z przykładowymi programami.

W bardzo obszernym rozdziale 8 znalazł się precyzyjny opis zmiennych systemowych, opis działania interpretera BASIC-a od strony kodu maszynowego, ilustrowany rysunkami opis organizacji pamięci i systemu operacyjnego ernal. Rozdział 9 poświęcony jest generacji dźwięku w C64, zaś 10 grafice, w tym również ruchomym obiektom — sprite.

Dwa ostatnie rozdziały zainteresują miłośników hardware. Można tu znaleźć schematy poszczególnych zespołów mikrokomputera C64, opisy wyprowadzeń układów scalonych, przebiegi czasowe sygnałów cyfrowych, opisy sterownika wizyjnego, generatora dźwięku, układu połączeń klawiatury, zasilacza, interfejsów, układów logicznych sterowania pamięcią itp.

Książkę zamykają schematy urządzeń peryferyjnych do samodzielnego wykonania: interfejsów Centronics, RS 232, modemu, programatora EPROM i dodatkowego modułu ROM (cartridge). Do każdego schematu dołączony jest opis oprogramowania wraz z listingami. Podsumowując można stwierdzić, że „Commodore za sva vremena nie odbiega poziomem od renomowanej literatury dotyczącej C64 wydawanej w USA, Wielkiej Brytanii i RFN, łącząc w jednej pozycji szeroki zestaw zagadnień związanych z tym najpopularniejszym na świecie komputerem domowym.

(jn)

PLOTER ROLAND DXY-980 A



Niezbędnym elementem wyposażenia każdego komputera osobistego wykorzystywanego do projektowania wspomaganego komputerowo (CAD — *Computer Aided Design*) jest ploter. Chcielibyśmy przedstawić Czytelnikom interesującą konstrukcję tego rodzaju: DXY-980A firmy Roland DG Corporation. Urządzenie to wyposażone jest w interfejsy: równoległy Centronics i szeregowy RS-232C i przystosowane jest do współpracy z kilkoma typami mikrokomputerów, przede wszystkim z IBM PC i Apple. Ploter może współpracować z typowym oprogramowaniem użytkowym IBM-a: programami komputerowego wspomagania projektowania Auto CAD i Versa CAD lub programami z dziedziny zastosowań ekonomicznych jak np. Lotus 1-2-3.

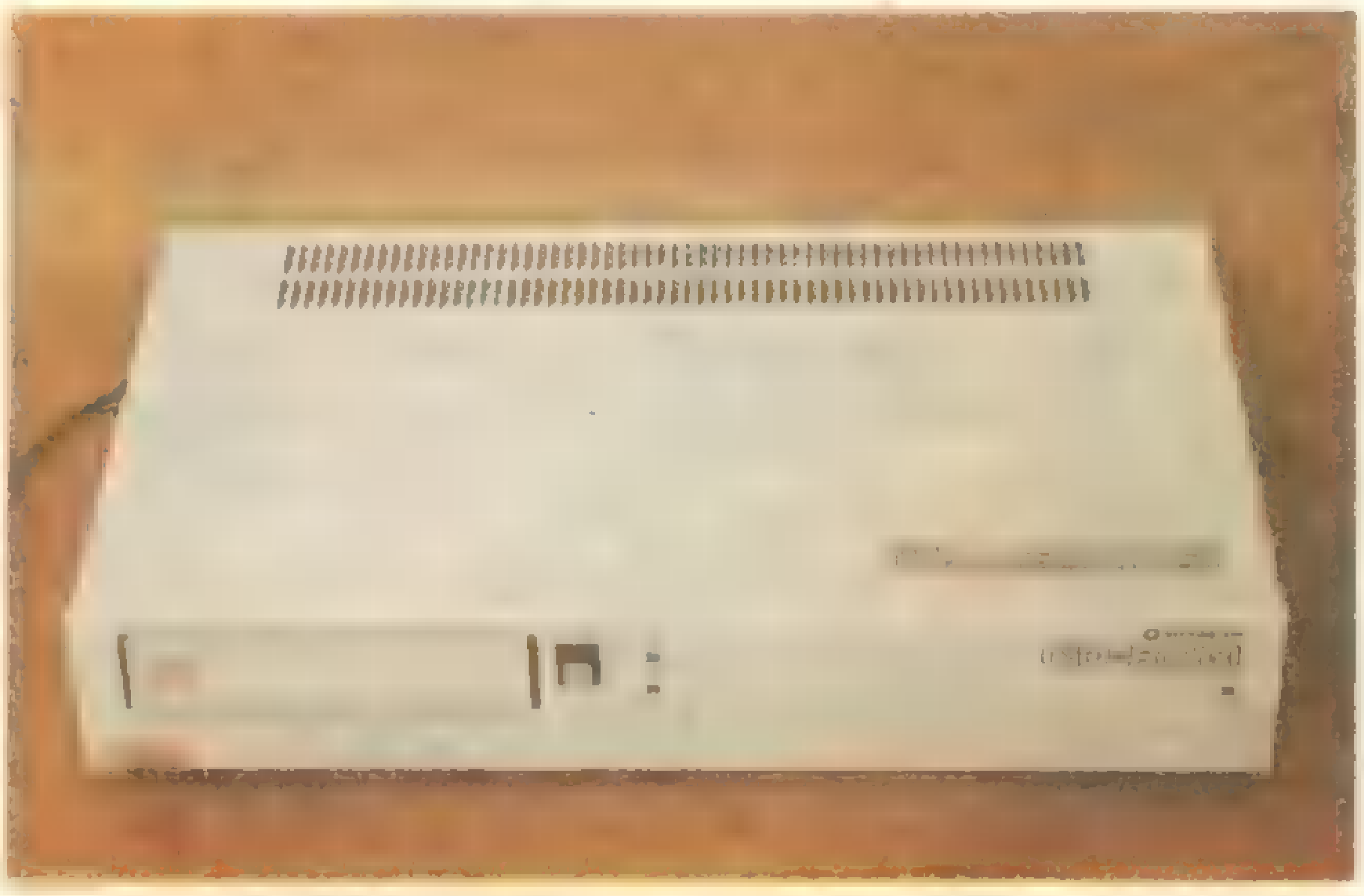
Kreślenie linii odbywa się z prędkością 230 mm/s (w dowolnym kierunku). Rozdzielczość rysunku wynosi 0,05 mm/krok. System „miękkiego lądowania" końcówki pisaka na papierze pozwolił na stosowanie tuszu kreślarskiego, a także poważnie zmniejszył hałaśliwość plotera. W przypadku, gdy końcówka pisząca zbyt długo spoczywa na papierze, jest ona automatycznie podnoszona do góry (tzw. *automatic pen-up device*), aby nie zrobił się kleks. Przycisk PAUSE pozwala na chwilowe wstrzymanie pracy urządzenia w dowolnym momencie. Arkusz papieru może mieć maksymalny format A3 (420 × 297 mm), podczas gdy obszar pracy końcówki plotera ma wymiary 380 × 297 mm. Papier jest „przysysany" elektrostatycznie do podłoża, tak aby nie robiły się na nim fałdki i nierówności.

Po lewej stronie „rysownicy" znajduje się magazynek 8 barwnych pisaków, pobieranych przez urządzenie automatycznie. Producent zaleca stosowanie własnych końcówek piszących (tzw. ceramicznych), ale zaznacza, że ploter jest przystosowany do używania rapidografów renomowanej zachodnioniemieckiej firmy Staedtler. Standardem są końcówki z włókienkiem (rapidograf), ale oprócz tego mogą być stosowane końcówki kuleczkowe (również na wodny tusz), jak i końcówki z włókienkiem na tusz olejowy.

W górnej części obudowy zainstalowano 11-przyciskową klawiaturę i dwa cyfrowe wyświetlacze na diodach LED informujących o aktualnej pozycji pisaka. Pojemność pamięci bufora wynosi 2,5 KB. Znaczne przyspieszenie współpracy z komputerem przynosi zastosowanie bufora danych SYA-350 będącego w istocie stacją dysku elastycznego 3,5 cala, do której transmitowane są z komputera dane przekazywane następnie do plotera.

Ploter ma wymiary 530 × 450 × 106 mm, masę 5,1 kg i może pracować zarówno w pozycji poziomej, jak i pochylony pod kątem 60°.

(jnkz)



PAMIĘCI KOMPUTEROWE COMPACT DISC JUŻ W PRODUKCJI

O masowych pamięciach optycznych poważnie zaczęto mówić zaledwie przed kilku laty, a już dziś pojawiają się na rynku pierwsze, seryjne modele tego sprzętu. Przykładem może być dyskowa pamięć stała Hitachi CDR-15028, bazująca na nośniku identycznym, jak w gramofonach laserowych typu Compact Disc: dyskach optycznych o średnicy 12 cm.

Urządzenie jest dostosowane do współpracy z komputerami klasy IBM PC/XT/AT. Do komputera należy włożyć pakiet specjalnego interfejsu i połączyć go przewodem z pamięcią dyskową. Zaletą nośnika typu CD jest duża odporność danych na uszkodzenia.

Całkowita pojemność pojedynczego dysku wynosi 552 MB, co odpowiada ok. 300 tysiącom stron maszynopisu (strona maszynopisu = ok. 1800 znaków). Standardowo stosowany w IBM PC/XT/AT system operacyjny MS-DOS/PC-DOS może jednak administrować pamięciami masowymi o pojemności maksymalnej co najwyżej 32 MB. Specjalne oprogramowanie, dostarczane wraz ze stacją, pozwala jednak podzielić dysk na segmenty po 32 MB i odwoływać się do każdego z nich jak do osobnej pamięci masowej.

Mimo krótkiego czasu, potrzebnego na zmianę ścieżki (1 ms) średni czas dostępu jest dość długi i wynosi ok. 1 s. Powierzchnia dysku jest bowiem podzielona aż na 20 000 ścieżek, co znacznie wydłuża czas potrzebny na odnalezienie właściwej ścieżki i dokładne pozycjonowanie głowicy. Transmisja danych odbywa się jednak szybko (176 KB/s). Prawdopodobieństwo wystąpienia błędu niemożliwego do skorygowania jest mniejsze od 10E-12.

Jak na razie, najpoważniejszym mankamentem nowej pamięci jest możliwość wyłącznie odczytywania danych. Praktyczne zastosowania ograniczają się zatem do przypadków, w których istnieje dostawca gotowych dysków z danymi. Przykładem może być archiwum obrazów graficznych o wielkiej rozdzielczości lub katalog wyrobów przemysłowych i ich producentów, liczący 186 000 pozycji, dostarczanych przez 60 000 wytwórców. Przy takiej liczbie danych krytycznym problemem stają się optymalne algorytmy przeszukiwania bazy danych.

Cena pamięci CDR-15028 wynosi ok. 2000 dolarów. Prawdziwej kariery urządzeń tego typu należy spodziewać się dopiero wtedy, gdy zintegrowane zostaną możliwości zarówno odczytu danych jak i dźwięku tak, by mogły one pełnić równocześnie funkcję elementu domowego zestawu Hi-Fi. Pod względem technicznym nic nie stoi tu na przeszkodzie. Inny kierunek rozwoju pamięci CD to możliwość zapisu danych we własnym zakresie, co pozwoliłoby uczynić z niego narzędzie do archiwowania danych. (rw)

SEMINARIUM „INFORMIKA"

ASEMBLER GENS 3

Część 3

Omawiając w poprzednim wykładzie budowę podstawowych zdań asemblera zajmowaliśmy się w istocie jednym z elementów składni, jakim są nazwy. Przypomnijmy, że poznane do tej pory nazwy możemy podzielić na dwie grupy:

1) nazwy zawarte w wewnętrznym słowniku asemblera (są nimi nazwy instrukcji, pseudoinstrukcji i dyrektyw),

2) nazwy utworzone przez programistę (nazwy te zostają umieszczone w Tablicy Symboli).

Wyrażenia arytmetyczne

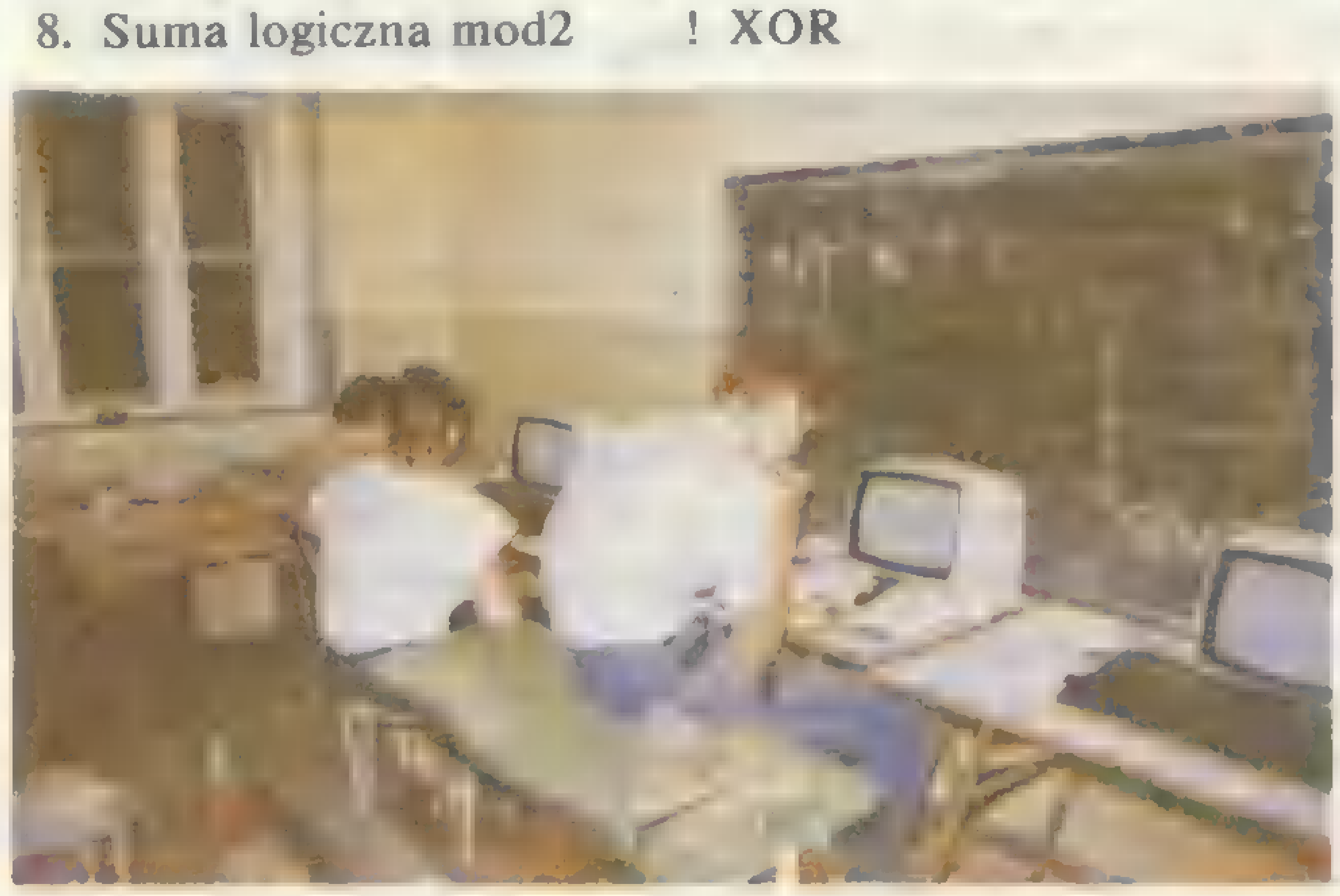
Drugim, funkcjonalnie odmiennym elementem składni asemblera są wyrażenia arytmetyczne. Wyjaśnijmy tu od razu, że występujące w polu argumentów stałe liczbowe są jedynie szczególnym przypadkiem argumentów akceptowanych przez asembler.

W ogólnym przypadku argument liczbowy może być zastąpiony wyrażeniem będącym złożeniem kilku operacji dwuargumentowych określonych w następującym zbiorze argumentów:

1. Stałe liczbowe tzn.
   1.1. Stałe dziesiętne (ciągi cyfr od zera do dziewięciu)
   .2. Stałe heksadecymalne (ciągi cyfr 0—9 lub liter A, B, C, D, E, F poprzedzone znakiem #)
   1.3. Stałe binarne (ciągi zer i jedynek poprzedzone znakiem %)
2. Stałe tekstowe (ciągi znaków ASCII rozpoczynające się i zakończone znakiem cudzysłowu)
3. Nazwy etykiet (adresy symboliczne)
4. Licznik Lokacji (oznaczany znakiem $)

Na wymienionych powyżej argumentach asembler dopuszcza wykonywanie następujących operacji arytmetycznych i logicznych:

1. Dodawanie	+
2. Odejmowanie	—
3. Mnożenie	*
4. Dzielenie całkowite	/
5. Modulo	? M?N $\overset{def}{=}$ M-(M N) N
6. Suma logiczna	@ OR
7. Iloczyn logiczny	& AND
8. Suma logiczna mod2	! XOR



Uwagi:

1. Należy pamiętać, że niezależnie od użytych przez nas wartości stałych dla asemblera będą istotne jedynie dwa najmniej znaczące bajty. Z założenia bowiem wszystkie stałe i wyrażenia z nich zbudowane będą obliczane modulo 65536. Ponadto jeśli w danej linii programowej wymagany będzie argument jednobajtowy (np. załadowanie obliczonego wyrażenia do pojedynczego rejestru) wartość wyrażenia zostanie obliczona modulo 256 (tzn. obcięta do najmniej znaczącego bajtu).

2. Błąd przekroczenia zakresu 65535 przy obliczaniu wyrażeń nie będzie sygnalizowany. Asembler wykryje natomiast próbę dzielenia przez wyrażenie równe zeru, oraz przekroczenie zakresu 32767 przy wykonywaniu operacji mnożenia. Drugi fakt wynika z tego, że przy obliczaniu wyrażeń liczby większe od 32767 interpretowane będą w kodzie uzupełnień jako liczby ujemne.

3. Określone powyżej operacje są z założenia dwuargumentowe, jednakże dopuszczalne jest i będzie poprawnie interpretowane użycie przed dowolnym z wymienionych argumentów znaku dodawania lub odejmowania.

4. Wyrażenia arytmetyczne są obliczane tylko w trakcie asemblacji, zatem w kodzie programu wynikowego zostaną umieszczone jedynie stałe. Nie należy więc oczekiwać, że zostaną im przypisane jakieś nowe instrukcje rozpoznawane przez mikroprocesor!

5. Wyrażenia arytmetyczne są obliczane kolejno od pierwszej do ostatniej linii programowej, nie należy więc używać w danej linii programowej wyrażenia, które nie zostało wcześniej zdefiniowane.

6. Wyrażenie arytmetyczne jest obliczane od lewej strony do prawej. Użycie nawiasów w celu zmiany kolejności operacji jest możliwe, jednakże jeśli wyrażenie jest argumentem instrukcji programowej użycie nawiasu spowoduje potraktowanie wyrażenia jako adresu pamięci.

Dokumentacja programu

Konsekwentne stosowanie wyrażeń arytmetycznych w istotny sposób zwiększa czytelność programu i powiększa szanse jego ewentualnych modyfikacji. Wybór właściwych nazw kojarzących z przeznaczeniem danych i działaniem procedur umożliwia uzyskanie programu samodokumentującego się. Zastosowanie podobnego zabiegu w wypadku stałych ma to dodatkowe uzasadnienie, że wszelkie tzw. oczywiste stałe są „oczywiste" przeważnie tylko dla twórcy programu i to przez krótki okres czasu. Natomiast praktyka programowania wykazuje, że okres modyfikowania każdego programu trwa zwykle od jego pierwszego uruchomienia do ostatniego zastosowania.

Drugim elementem zasadniczo wpływającym na czytelność programu są komentarze. Ich brak jest na ogół wynikiem mylnego przekonania programującego o ostateczności i jednorazowości zastosowania danego programu. Lista rozkazowa procesora Z80 umożliwia tworzenie bardzo efektywnego i zwartego kodu. Częstokroć sekwencja czterech do ośmiu linii programowych może realizować dość skomplikowaną funkcję wymagającą odpowiedniego komentarza. Jeśli więc szanujemy własny czas dokumentujmy każdą sprawdzoną procedurę z myślą o jej wykorzystaniu w innych programach.

Od problemu do programu

Omawiane problemy zilustrujemy krótkim przykładem, w którym wykorzystamy wyrażenia arytmetyczne. Załóżmy, że dla rozwiązania pewnego problemu niezbędne jest wykorzystanie tablicy dwuwymiarowej wypełnionej danymi jednobajtowymi. Zadaniem naszym będzie obliczenie sumy elementów poszczególnych wierszy i umieszczenie dwubajtowego wyniku w dwóch ostatnich kolumnach danego wiersza. Załóżmy, że danymi będą wielkości opadów deszczu mierzone w milimetrach rano i wieczorem przez siedem kolejnych dni tygodnia. Jeśli założymy dodatkowo, że z prośbą o napisanie programu opracowującego takie dane zwrócił się do nas znajomy meteorolog amator, to natychmiastowe kodowanie programu z takimi założeniami powinno być ostatnim pomysłem, który przyjdzie nam do głowy. Nawet zakładając, że prawdopodobnie nie zechcemy mierzyć opadów dodatkowo w południe, możemy przypuścić co najmniej możliwość opadów przez dłuższy okres czasu. Ponadto może okazać się konieczne obliczanie sum dziennych opadów. Dopiero po przemyśleniu przynajmniej kilku aspektów problemu (a nawet w tak prostym zadaniu widać, że lista wątpliwości jest znacznie dłuższa) rozpoczynamy planowanie struktury danych a następnie programu. Zbudowana nieco na wyrost tabela przedstawia się następująco:

	Pn	Wt	Sr	Cz	Pt	So	Nd	2 bajty
Rano	X	X	X	X	X	X	X	SUMA 1
Wiecz.	X	X	X	X	X	X	X	SUMA 2
								SUMA 3
	1 bajt							

Ponieważ najważniejszym zadaniem naszego programu jest obliczanie sumy elementów w następnym etapie obmyślamy i kodujemy odpowiednią procedurę. Jeśli nasza procedura ma przydać się jeszcze w innych programach, to sprecyzujemy najpierw zakres jej zastosowania oraz sposób przekazywania parametrów do procedury i pobierania wyników. W tym przypadku wystarczy wykorzystać do tego celu rejestry procesora. Ich wybór nie jest na ogół przypadkowy co jest powodowane pewną specjalizacją ich wykorzystywania w liście rozkazowej. Jeśli wyjdziemy z założenia, że do adresowania pamięci najlepiej nadaje się para rejestrów HL, to do przechowywania wyniku częściowego przeznaczamy parę rejestrów DE. Powodowane jest to możliwością łatwej wymiany zawartości wymienionych par. W konsekwencji na licznik długości sumowanego obszaru pozostaje nam rejestr A lub para BC. Oczywiście przyjmując inne założenia możemy dojść do zupełnie innego rozwiązania.

Przytoczenie tego tylko pozornie niepoważnego przykładu miało na celu przybliżenie czytelnikowi kilku zasadniczych problemów programowania w języku asemblera. Ich zrozumienie ma zasadnicze znaczenie już na tym etapie wykładu. Przyjęte powyżej rozwiązanie choć wymagało nieco większego nakładu, ma jedną zasadniczą zaletę: łatwość modyfikacji programu. Jeśli nasz program miałby w przyszłości opracować większą liczbę danych

jedyną konieczną modyfikacją będzie zmiana deklaracji liczby dni w linii 40 programu źródłowego. Istotnym ograniczeniem dla poprawnego działania procedury sumowania jest przyjęcie liczby większej od 255. Oczywiście można napisać ogólniejszą procedurę i bardziej uniwersalny program. Niestety jest to okupione w pierwszym przypadku na ogół dłuższym czasem działania programu, a w drugim wydłużeniem czasu opracowania i przetestowania programu. O ile pierwszy przypadek nie jest na ogół drastyczny, to w drugim wybranie rozsądnego kompromisu jest konieczne. Nie trzeba nikogo przekonywać, że działający poprawnie program jest więcej wart niż najoryginalniejsza koncepcja istniejąca wyłącznie w wyobraźni programisty.

Znaczenie programu źródłowego

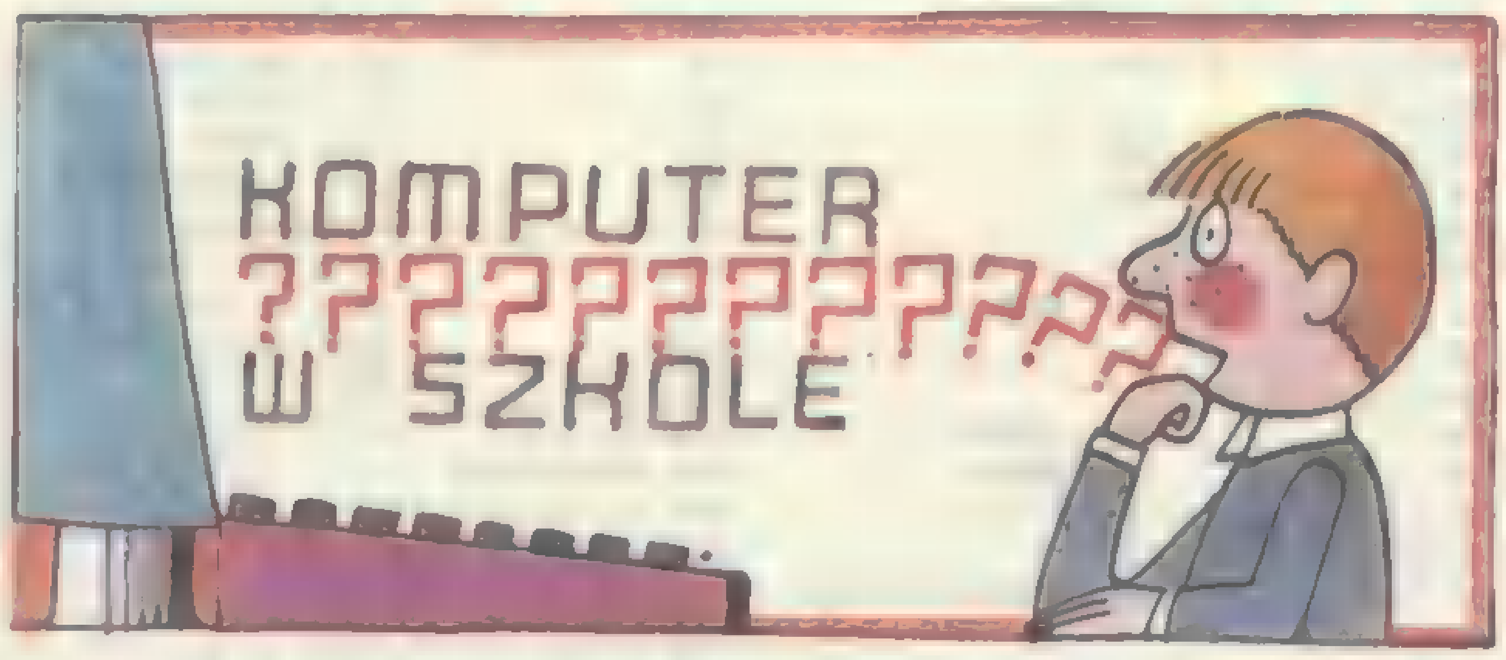
Należy zdać sobie sprawę z tego, że wykorzystana w programie przykładowym tablica dwuwymiarowa jest obiektem abstrakcyjnym, istniejącym jedynie w dokumentacji programu źródłowego. Z punktu widzenia mikroprocesora realizującego program wynikowy istnieje jedynie pewien ciągły obszar przestrzeni adresowej. Ponieważ nie istnieją żadne oczywiste mechanizmy pozwalające procesorowi ustalić strukturę danych, a w ogólnym przypadku odróżnić dane od programu, jedynym obiektem zawierającym całkowitą informację jest program źródłowy. Ponieważ w dalszym ciągu wykładu zajmiemy się asemblacją programu prowadzącą do otrzymania programu wynikowego, nie należy sądzić, że na jakimkolwiek etapie będziemy mogli zrezygnować z posiadania programu źródłowego. Dość często praktykowane publikowanie programów wyłącznie w postaci kodu wynikowego dowodzi jedynie niepoważnego traktowania problemu.

Tadeusz Basista

```
10  [illegible] "TABLICA"
20  ;-----------------
30  ;STALE
40  DNI          EQU  7
50  POMIAR       EQU  2
60  ;
70  WIERSZ       EQU  POMIAR+1
80  KOLUMN       EQU  DNI+2
90  ;------------------------
100 ;LOKACJA
110              ORG  60000
120              JP   START
130 ;------------------------
140 ;REZERWACJA TABLICY
150 TABL         DEFS WIERSZ*KOLUMN
160 ;------------------------
170 ;WYLICZENIE ADRESOW [illegible]
[illegible] [illegible]  EQU  1*KOLUMN+TABL-2
190 [illegible]  EQU  2*KOLUMN+TABL-2
[illegible] [illegible]  [illegible]  3*KOLUMN+TABL-2
210 ;------------------------
[illegible] ;[illegible] GLOWNY
[illegible] ;------------------------
240 START        LD   HL,TABL
250              LD   [illegible]
260              CALL SUMUJ
270              LD   (SUMA1),DE
280 ;teraz drugi wiersz
290              LD   HL,TABL+KOLUMN
300              LD   A,DNI
[illegible]      CALL [illegible]
320              LD   (SUMA2),DE
330 ;i [illegible] koncu obydwie [illegible]
340              LD   HL,(SUMA1)
350              ADD  HL,DE
360              LD   (SUMA3),HL
370              RET
380 ;------------------------
390 ;PROCEDURA
400 ;------------------------
410 ;A -jest licznikiem
420 ;HL-adres elementu
[illegible] ;DE-aktualny wynik
[illegible] [illegible]  LD   DE,0
450              LD   B,0
[illegible] [illegible]  LD   C,(HL)
470              [illegible] DE,HL
480              ADD  HL,BC
490              [illegible] DE,HL
500              INC  HL
[illegible]      [illegible] A
520              JR   NZ,DALEJ
530              RET
540 ;------------------------
```





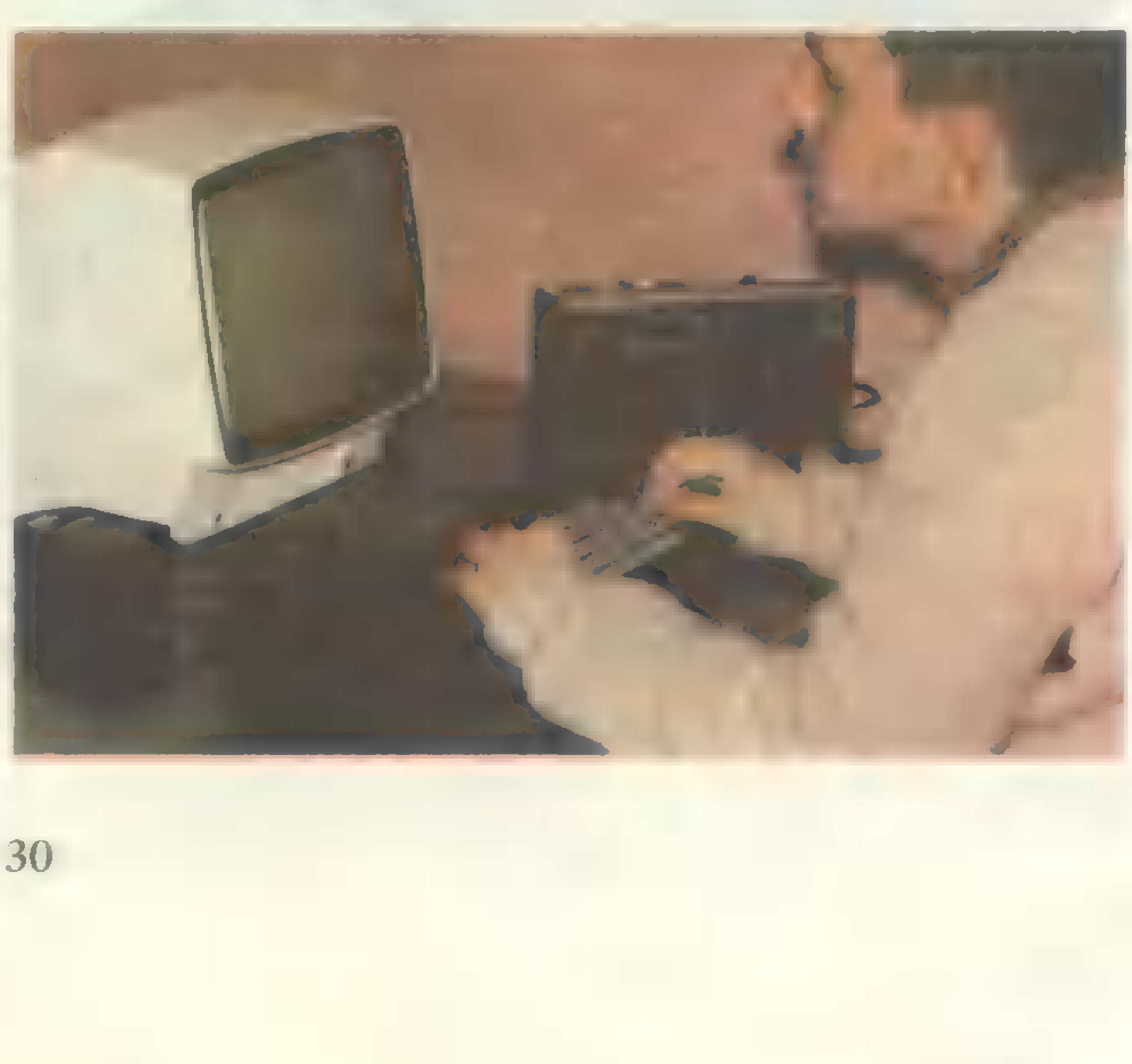


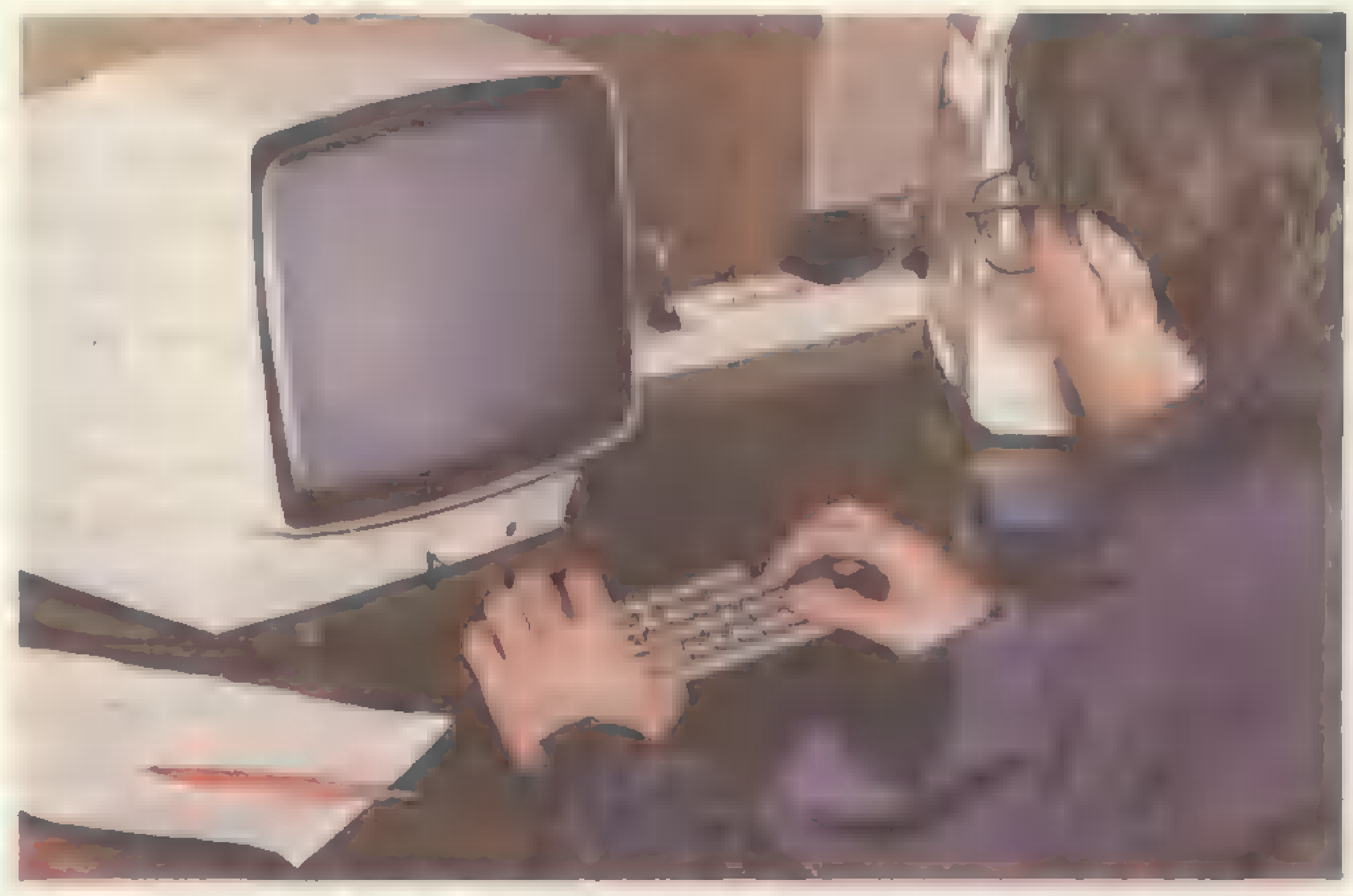
OD POMYSŁU DO PROGRAMU

Każdy, kto napisał choć jeden większy program, doskonale zdaje sobie sprawę, jak długą drogę należy pokonać od początkowego pomysłu i wstępnego wyobrażenia o funkcji i formie programu, aż po uzyskanie jego realnego kształtu i niezawodnego funkcjonowania. Na ogół nieuniknione są dziesiątki, a przy programach bardziej złożonych nawet setki godzin pracy. Czy istnieje możliwość skrócenia tego czasu? Zapewne tak. Można przecież stosować różnego typu programy użytkowe, mające za zadanie ułatwiać redakcję tworzonego programu, lub choćby jego części (np. rysunku). Można również stosować zaczerpnięte z innego źródła gotowe bloki programowe, czy też całe procedury. W tych jednak przypadkach zysk czasowy może okazać się złudny. Samo wyszukiwanie odpowiednich schematów, a następnie ich adaptacja dla potrzeb programu pochłania przecież sporo czasu.

Inną możliwością jest metoda stosowana przez duże firmy software'owe, którym zależy na szybkim ukazywaniu się ich programów. Metoda nie jest odkrywcza. Po prostu nad programem pracują duże zespoły ludzi, złożone z kilkudziesięciu osób. Zapewne liczba „roboczogodzin" nie ulega w ten sposób istotnej zmianie, bo zwiększa się o czas potrzebny na wzajemne konsultacje, a zmniejsza, z uwagi na dużą efektywność pracy, wynikającą z podziału zadań wśród członków zespołu. Każdy opracowuje bowiem jedynie wąski temat, za to techniką, w której specjalizuje się. Zysk na czasie jest wówczas oczywisty.

Wojtek Sieroń z Mielca przy pracy nad programem





Mirosław Furmanek, jeden z najmłodszych uczestników obozu reprezentujący „InforMik" Knurów — przy komputerze

Czterem uczniom, spośród uczestników dwutygodniowego obozu mikrokomputerowego, który odbył się wiosną bieżącego roku w Centralnej Szkole Instruktorów Harcerskich w Załęczu Wielkim (woj. sieradzkie), zaproponowano, by właśnie tą metodą opracowali program o charakterze edukacyjnym. Czasu na realizację tego zamierzenia nie było wiele — zaledwie kilka dni. Wystarczyło to jednak, by powstał program w miarę efektowny i o dużych możliwościach.

Prześledźmy teraz proces jego powstawania, skupiając na razie uwagę nie na samej tematyce programu, lecz na metodyce jego tworzenia. Jest oczywiste, że przed powołaniem zespołu, osoba prowadząca powinna mieć wypracowaną, w miarę pełną, koncepcję ostatecznego kształtu programu, powinna znać jego schemat blokowy i zdawać sobie sprawę, w jaki sposób najlepiej będzie realizować poszczególne bloki. W konkretnym przypadku omawianego programu, wiadome było, że będzie on wymagał przygotowania dość złożonego algorytmu matematycznego, przeprowadzania prostej animacji, tablicowania niezbędnych danych oraz opracowania obsługi całego programu i nadania mu szaty graficznej.

Kluczowym problemem dla całego przedsięwzięcia wydaje się być optymalny przydział zadań poszczególnym członkom zespołu. Ważne jest przy tym posiadanie pełnego rozeznania w merytorycznej wiedzy uczniów, związanej z tematem programu oraz znajomość ich stopnia zaawansowania w różnych technikach programowania. Korzystając z możliwości indywidualnego przydziału zadań, blok związany np. z animacją powinien otrzymać uczeń znający kod maszynowy mikroprocesora, a opracowanie algorytmu obliczeniowego — uczeń dobrze orientujący się w części merytorycznej zagadnienia. Możliwości jest wiele, ważne jest jednak, by cały zespół znał całość postawionych przed nim zadań, orientował się w zaawansowaniu prac poszczególnych jego członków, by po prostu współpracował ze sobą. Praktyka wskazuje, iż młodzież wnosi wiele własnej inwencji, niejednokrotnie modyfikując założenia wstępne, a wzajemne konsultacje i naturalny doping zwiększają tylko efektywność prac.

Z uwag praktycznych warto jeszcze zaznaczyć, że podziałowi zadań wśród członków zespołu powinien towarzyszyć podział miejsca w pamięci komputera, tzn. podział nazw zmiennych, numerów wierszy w programie i adresów obszarów w pamięci mikrokomputera. Ułatwi

to późniejsze „sklejanie" poszczególnych bloków w jedną całość.

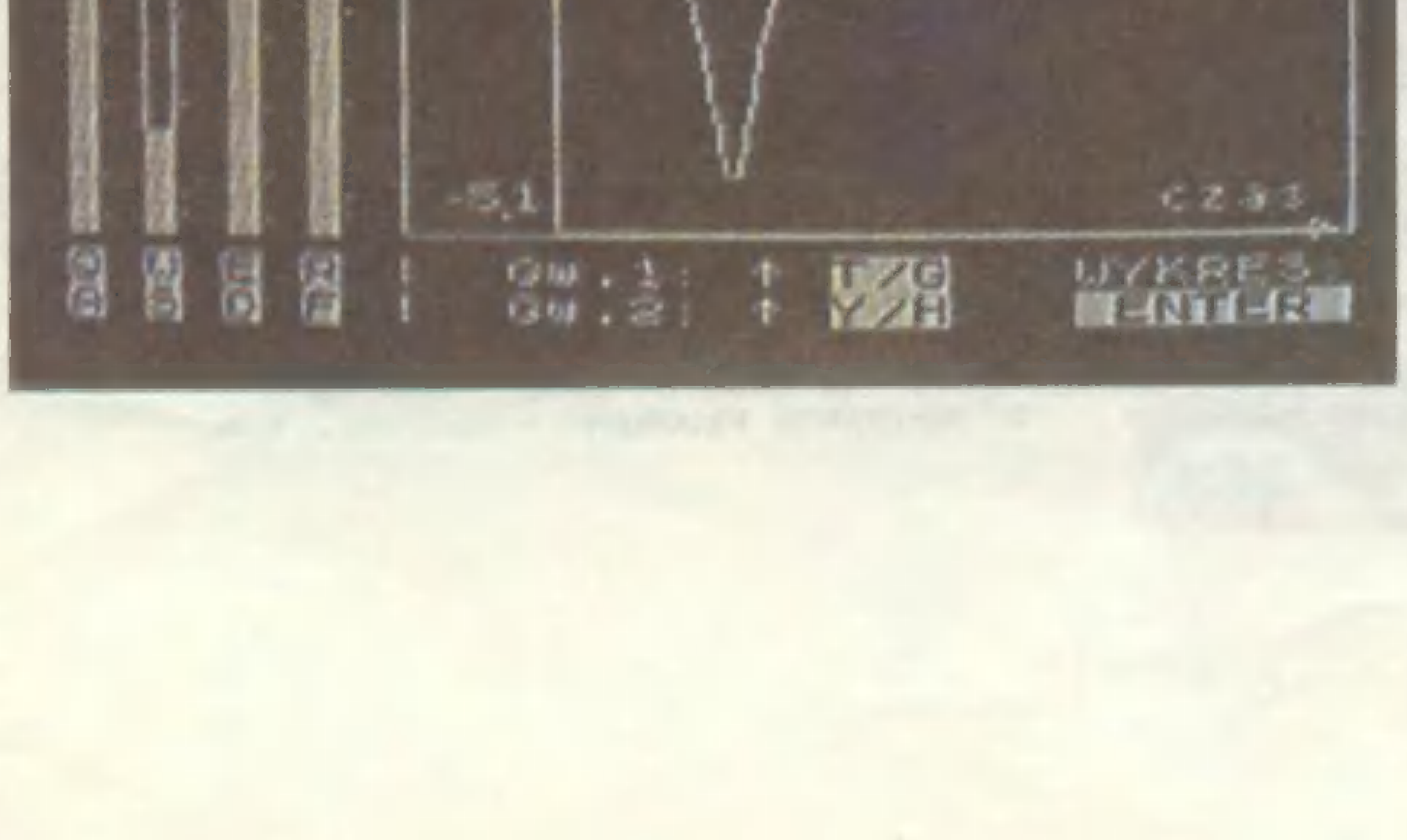
Powróćmy teraz do programu, wykonanego przez młodzież w Załęczu. Jego listing zamieszczamy obok. Z założenia program ten miał mieć charakter edukacyjny co narzuciło, praktycznie bez możliwości wyboru, typ mikrokomputera, na który został napisany — ZX Spectrum. Wybrano temat astronomiczny: gwiazdy zmienne zaćmieniowe, z którego można korzystać na lekcjach fizyki przy omawianiu gwiazd i zagadnień związanych z fotometrią gwiazdową.

Wiadomo, że ponad połowa gwiazd tworzy układy podwójne (część z nich nawet wielokrotne). Układy podwójne to takie, w których dwie gwiazdy obiegają wspólny środek masy. W niektórych przypadkach ich orbity są tak usytuowane w przestrzeni względem obserwatora, iż może dochodzić do wzajemnego przesłaniania się składników. Obserwator dostrzega wówczas zmianę jasności całego układu. Wykres przedstawiający zmianę tej jasności w czasie nosi nazwę krzywej jasności. Charakter tej krzywej zależy od parametrów fizycznych gwiazd, głównie od ich temperatur powierzchniowych i promieni, a także od usytuowania orbity w przestrzeni (gwiazdy nie muszą przesłaniać się centralnie). Różnorodność przypadków realizowanych przez przyrodę jest ogromna. Celem omawianego programu jest umożliwienie zaprezentowania dowolnego z tych przypadków.

Program obsługuje się bardzo łatwo poprzez naciskanie odpowiednich klawiszy. I tak, dla pierwszej z gwiazd można zmieniać jej rozmiary (klawisz Q lub A), temperaturę (W lub S) oraz położenie środka (T lub G). W analogiczny sposób można zmieniać parametry drugiej z gwiazd naciskając odpowiednio klawisze E lub D, R lub F oraz Y lub H. Po doprowadzeniu do interesującego nas przypadku wystarczy nacisnąć klawisz ENTER, a wówczas na ekranie rysowana będzie krzywa jasności (wyskalowana w absolutnych wielkościach gwiazdowych) i jednocześnie przedstawiany będzie przebieg zjawiska. Cykl ten można w każdej chwili przerwać naciskając klawisz P, stworzy to możliwość wyboru nowych parametrów.

Autorzy życzą przyjemnej nauki.

Opracował:
Marek T. Szczepański




```
  1 REM       GWIAZDY ZMIENNE - ZACMIENIOWE
  5 GO TO 8100: REM
           PRZYGOTOWANIE  PLANSZY

 10 LET pp=500: LET v=0: LET h=256: LET q=60000: LET w=6051
8: LET e=60582: LET u=w-21: BORDER 0: PAPER 0: INK 7: CLS
 15 DIM v(14)
 20 FOR n=1 TO 14: READ v(n)
 25 NEXT n: DATA 1,65,2,66,3,67,4,5,68,6,69,7,70,71: REM
   ODCIENIE SZAROSCI DLA MONITORA CZARNO-BIAŁEGO
 30 PRINT #0; INVERSE 1;"Q"; INVERSE 0;" "; INVERSE 1;"W";
INVERSE 0;" "; INVERSE 1;"E"; INVERSE 0;" "; INVERSE 1;"R";
INVERSE 0;" ¦ Y gw.1: "; INVERSE 1;"T/G"; INVERSE 0;"  ¦ WYK
RES:"
 35 PRINT #1; INVERSE 1;"A"; INVERSE 0;" "; INVERSE 1;"S";
INVERSE 0;" "; INVERSE 1;"D"; INVERSE 0;" "; INVERSE 1;"F";
INVERSE 0;" ¦ Y gw.2: "; INVERSE 1;"Y/H"; INVERSE 0;"  ¦ ";
INVERSE 1;" ENTER "
 40 LET s0=LN 10: LET s1=11.2764: LET s2=101788: LET Ms=4.8
0001: LET r=55.0001: LET t=33: LET rr=67.0001: LET tt=44
 50 LET y1=136: LET y2=143: LET x1=31: LET x2=32: POKE u+6,
x1: POKE u+14,x2
 60 LET r1=18: LET t1=32: LET r2=22: LET t2=43: POKE 23658,
8: LET vv=0: GO TO 200
105 LET f=USR 60447
110 POKE u-7,PEEK (u+2): POKE u-6,PEEK (u+3): POKE 60229,20
0: POKE 60104,42: POKE 60105,79: POKE u-5,x1: POKE u-4,y1
130 POKE u-3,x1: POKE u-2,PEEK u: POKE u-1,PEEK (u+1): LET
f=v(n1)>9: INK (v(n1)-64*f): BRIGHT f: LET f=USR 60104: GO T
O 170
140 POKE u-7,PEEK (u+10): POKE u-6,PEEK (u+11): POKE 60229,
96: POKE 60104,24: POKE 60105,102: POKE u-5,x2: POKE u-4,y2
160 POKE u-3,x2: POKE u-2,PEEK (u+8): POKE u-1,PEEK (u+9):
LET f=v(n2)>9: INK (v(n2)-64*f): BRIGHT f: LET f=USR 60000
170 POKE 60229,96: POKE 60104,42: POKE 60105,79: IF v=1 THE
N  GO SUB 810: LET f=USR 60232: LET f=USR 60424
175 LET v=1: INK 7: BRIGHT 0: INVERSE 0: RETURN
200 FOR m=1 TO 6: READ a,b,c,d
210     FOR n=d TO c STEP -1: PLOT a,n: PLOT b,n
220     NEXT n
230     FOR n=a TO b: PLOT n,c: PLOT n,d
240     NEXT n
250 NEXT m
260 DATA 68,255,111,175,68,255,3,107,0,6,3,91,16,22,3,91,33
,39,3,91,49,55,3,91
270 FOR n=4 TO 33: PLOT 1,n: DRAW 4,0: PLOT 17,n: DRAW 4,0:
     PLOT 34,n: DRAW 4,0: PLOT 50,n: DRAW 4,0
275 NEXT n
280 FOR n=34 TO 44: PLOT 1,n: DRAW 4,0: PLOT 34,n: DRAW 4,0
    : PLOT 50,n: DRAW 4,0
285 NEXT n
290 FOR n=45 TO 55: PLOT 1,n: DRAW 4,0: PLOT 34,n: DRAW 4,0
295 NEXT n
300 FOR n=56 TO 67: PLOT 34,n: DRAW 4,0
305 NEXT n
310 PRINT AT 9,0;"R T R T": PLOT  INVERSE 1;7,94: GO SUB 80
49: PLOT  INVERSE 1;20,94: GO SUB 8049: PLOT  INVERSE 1;40,9
4: GO SUB 8050: PLOT  INVERSE 1;53,94: GO SUB 8050
320 FOR n=15 TO 90 STEP 15: PLOT 8,n: PLOT 41,n
330 NEXT n
340 FOR n=11 TO 91 STEP 20: PLOT 24,n: PLOT 57,n
350 NEXT n: GO SUB 840: GO SUB 860: POKE 60440,226: LET f=U
SR 60424: POKE 60440,218: LET f=USR 60424
360 GO SUB 880: LET v=0: GO SUB 110: GO SUB 140
400 INK 6: RESTORE 425: GO SUB 405: PAUSE 0: INK 7: RESTORE
425: GO SUB 405: GO TO 435
405 PRINT AT 2,14;"G W I A Z D Y";AT 4,14;"Z M I E N N E";A
T 6,10;"Z A C M I E N I O W E"
410 PRINT AT 10,12;"Program wykonali:";AT 12,10;"Wojciech S
ieron";AT 13,10;"Andrzej Strzeszewski";AT 14,10;"Miroslaw Fu
rmanek";AT 15,10;"Adam Takiel"
415 PRINT AT 17,13;"pod opieka";AT 18,10;"M.T.Szczepanskieg
o";AT 20,10;"Zalecze,kwiecien 1987"
420 FOR n=1 TO 12: READ a,b: PLOT a,b
425 NEXT n: DATA 195,78,132,48,180,32,171,30,98,11,100,13,1
22,59,124,61,108,8,203,14,116,128,117,129
430 RETURN
440 LET v=0: GO SUB 800
450 LET d$="R =    *R": PRINT AT 1,9;d$;TAB 22;d$: LET x=14
4: GO SUB 960: LET x=247: GO SUB 960
455 LET d$="T =     K": PRINT AT 3,9;d$;TAB 22;d$: LET x=12
0: GO SUB 950: LET x=224: GO SUB 950
460 LET D$="Y =": PRINT AT 5,9;d$;TAB 22;d$
470 FOR m=79 TO 184 STEP 105
475     FOR n=126 TO 158 STEP 16: PLOT  INVERSE 1;m,n:
          GO SUB 8049+(m=184)
480     NEXT n
485 NEXT m: GO SUB 970
495 PRINT AT 7,13; INVERSE 1;" WYBIERZ DANE ": REM

              WYBIERANIE DANYCH

500 IF INKEY$="" THEN  GO TO pp
505 LET a=CODE INKEY$: BEEP .1,40: GO SUB 810: LET f=USR 60
460
510 IF a=65 OR a=81 THEN  GO TO 600
515 IF a=83 OR a=87 THEN  GO TO 640
520 IF a=68 OR a=69 THEN  GO TO 620
525 IF a=70 OR a=82 THEN  GO TO 660
530 IF a=71 OR a=84 THEN  GO TO 680
535 IF a=72 OR a=89 THEN  GO TO 690
540 IF a=13 THEN  GO SUB 810: LET f=USR 60424: GO TO 999
545 GO TO pp
600 IF (r1+y1>173 OR y1-r1<114) AND a=81 THEN  GO TO pp
605 IF r<5 AND a=65 THEN  GO TO pp
610 LET r=r+(a=81): PLOT  OVER (a=65);1,r: DRAW  OVER (a=65
```

```
);4,0: LET r=r-(a=65): GO SUB 980: IF r1=INT (r/3) THEN  GO
TO pp
 615 LET r1=INT (r/3): GO TO 820
 620 IF (r2+y2>173 OR y2-r2<114) AND a=69 THEN  GO TO pp
 625 IF rr<5 AND a=68 THEN  GO TO pp
 630> LET rr=rr+(a=69): PLOT  OVER (a=68);34,rr: DRAW  OVER
(a=68);4,0: LET rr=rr-(a=68): GO SUB 982: IF r2=INT (rr/3) T
HEN  GO TO pp
 635 LET r2=INT (rr/3): GO TO 820
 640 IF t=4 AND a=83 THEN  GO TO pp
 645 IF t=90 AND a=87 THEN  GO TO pp
 650 LET t=t+(a=87): PLOT  OVER (a=83);17,t: DRAW  OVER (a=8
3);4,0: LET t=t-(a=83): LET t1=t-1: GO SUB 984: IF n1=INT ((
t1+9)/7) THEN  GO TO pp
 655 GO SUB 840: GO TO pp
 660 IF tt=4 AND a=70 THEN  GO TO pp
 665 IF tt=90 AND a=82 THEN  GO TO pp
 670 LET tt=tt+(a=82): PLOT  OVER (a=70);50,tt: DRAW  OVER (
a=70);4,0: LET tt=tt-(a=70): LET t2=tt-1: GO SUB 986: IF n2=
INT ((t2+9)/7) THEN  GO TO pp
 675 GO SUB 860: GO TO pp
 680 IF y1+r1=173 AND a=84 OR y1-r1=113 AND a=71 THEN  GO TO
 pp
 685 LET y1=y1-(a=71)+(a=84): GO SUB 988: GO TO 820
 690 IF y2+r2=173 AND a=89 OR y2-r2=113 AND a=72 THEN  GO TO
 pp
 695 LET y2=y2-(a=72)+(a=89): GO SUB 990: GO TO 820: REM

                 PODPROGRAMY

 700 LET ee=SQR ((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2)): IF ee>=IN
T r1+INT r2 THEN  LET Mw=Max: RETURN
 710 IF ee<=ABS (r1-r2) THEN  LET Sn=pole: GO TO 740
 720 LET kk=(rk1-rk2+ee*ee)/2/ee: LET a1=ACS (kk/r1): LET a2
=ACS ((ee-kk)/r2)
 730 LET Sn=a1*rk1-r1*kk*SIN a1+a2*rk2-r2*(ee-kk)*SIN a2
 740 LET Mw=M2-2.5*LN ((s6-Sn*tk2)/PI/rk2/tk2)/s0: RETURN
 750 LET v=2: LET f=t1: LET t1=t2: LET t2=f: LET f=286-y1: L
ET y1=286-y2: LET y2=f: LET f=USR 60435
 760 LET f=rk1: LET rk1=rk2: LET rk2=f: LET f=tk1: LET tk1=t
k2: LET tk2=f: LET f=M1: LET M1=M2: LET M2=f: LET f=r1: LET
r1=r2: LET r2=f: IF ee=0 THEN  RETURN
 770 LET kk=(rk1-rk2+ee*ee)/2/ee: RETURN
 800 POKE 60237,64+8*(v=3)+154*(v=4): POKE 60234,194-8*(v=4)
+8*(v=3): POKE 60240,24-16*(v=4)-8*(v=3): LET f=USR 60232: P
OKE 60234,218: POKE 60237,64: POKE 60240,8: POKE 60432,24: R
ETURN
 810 POKE 60440,186: POKE 60432,8: LET f=USR 60424: POKE 604
40,218: POKE 60432,24: RETURN
 820 GO SUB 880: LET v=0: GO SUB 110: GO SUB 140: GO TO pp
 840 LET n1=INT ((t1+9)/7): FOR n=0 TO 7: FOR m=0 TO 3: POKE
 22528+32*n+m,v(n1): NEXT m: NEXT n: INK 7: BRIGHT 0: RETURN

 860 LET n2=INT ((t2+9)/7): FOR n=0 TO 7: FOR m=4 TO 7: POKE
 22528+32*n+m,v(n2): NEXT m: NEXT n: INK 7: BRIGHT 0: RETURN

 880 RESTORE 910
 885 FOR m=1 TO 4: READ a,b: LET k=PEEK (w-1+2*b)+h*PEEK (w+
2*b): LET p=PEEK (w-3+2*a)+h*PEEK (w-2+2*a)
 890 LET f=e-1+2*k: POKE u+8*(m>2),f-h*INT (f/h): POKE u+1+8
*(m>2),INT (f/h): LET f=k-p: POKE u+2+2*(m=2 OR m=4)+8*(m>2)
,f-h*INT (f/h): POKE u+3+2*(m=2 OR m=4)+8*(m>2),INT (f/h)
 900 NEXT m: RETURN
 910 DATA 0,INT r1+1-(vv=0),INT r1+1-(vv=0),INT r1+1-(vv=0),
0,INT r2+1-(vv=0),INT r2+1-(vv=0),INT r2+1-(vv=0)
 930 IF t1<t2 THEN  LET f=USR 60244: LET f=USR 60283: GO TO
948
 935 IF t1=t2 AND n1<4 THEN  LET f=USR 60322: LET f=USR 6028
3: GO TO 948
 940 IF t1=t2 THEN  LET f=USR 60244: LET f=USR 60361: GO TO
948
 945 LET f=USR 60322: LET f=USR 60361
 948 LET f=USR 60232: RETURN
 950 LET y=145: PLOT  INVERSE 1;x,y: GO SUB 8049: PLOT  INVE
RSE 1;x+5,y: GO SUB 8048: PLOT  INVERSE 1;x+10,y+2: GO SUB 8
051: RETURN
 960 LET y=158: PLOT  INVERSE 1;x,y: GO SUB 8053: PLOT  INVE
RSE 1;x+5,y+1: GO SUB 8076: RETURN
 970 FOR n=980 TO 990 STEP 2: GO SUB n
 975 NEXT n: RETURN
 980 LET d$=STR$ (r/15): PRINT AT 1,12;d$( TO 4): GO TO 992
 982 LET d$=STR$ (rr/15): PRINT AT 1,25;d$( TO 4): GO TO 992
 984 PRINT AT 3,12;t1;TAB 14;"*": GO TO 992
 986 PRINT AT 3,25;t2;TAB 27;"*": GO TO 992
 988 PRINT AT 5,12;y1-143;" ": GO TO 992
 990 PRINT AT 5,25;y2-143;" "
 992 GO SUB 810: LET f=USR 60460: LET v=4: GO SUB 800: RETUR
N
 995 REM
                 PROGRAM ZASADNICZY

1000 LET v=0: GO SUB 105: GO SUB 140: POKE 60432,8
1005 LET x1=x2: LET r1=r/3: LET r2=rr/3: LET rk1=r1*r1: LET
tk1=t1^4: LET rk2=r2*r2: LET tk2=t2^4: LET s6=PI*(rk1*tk1+rk
2*tk2)
1010 LET vv=1: LET v=1: IF ABS (y2-y1)>=INT r1+INT r2 THEN
LET d=INT ((r1+r2+7)/2): GO TO 1020
1015 LET d=r1+r2+4: LET a1=ASN ((y2-y1)/d): LET d=INT (.5+d*
COS a1): LET b=d/4-INT (d/4): LET d=(d+8-b*4)/2
1020 LET s=98: PLOT s,7: DRAW 0,s: DRAW -2,-3: DRAW 4,0: DRA
W -2,3: PLOT s,7: DRAW 154,0: DRAW -3,2: DRAW 0,-4: DRAW 3,2
1025 PRINT AT 20,27;"czas";AT 9,13;"M": IF r1>r2 THEN  LET p
ole=PI*rk2: GO TO 1035
1030 LET pole=PI*rk1
1035 LET M1=Ms-10*LN t1/s0-5*LN r1/s0+s1: LET M2=Ms-10*LN t2
/s0-5*LN r2/s0+s1: LET Max=M1-2.5*LN (1+10^(.4*(M1-M2)))/s0
1040 GO SUB 700: LET Min=Mw: GO SUB 760: GO SUB 700: GO SUB
760
```

```
1050 LET d$=STR$ Max: LET x=70: LET y=93: GO SUB 8020: IF Mi
n>Mw THEN  GO TO 1070
1060 LET f=Min: LET Min=Mw: LET Mw=f
1070 LET dM=Min-Max: IF Max=Min THEN  GO TO 1140
1080 LET d$=STR$ Min: LET y=13: GO SUB 8020: LET d$=STR$ Mw:
 LET y=INT (93.5+(Max-Mw)*80/dM): IF y>87 THEN  LET y=87
1090 IF t1=t2 THEN  GO TO 1140
1100 IF y<19 THEN  LET y=19
1110 GO SUB 8020
1140 LET j=95: LET z=s
1150 GO SUB 880: POKE u+6.160-d: POKE u+7,y1: POKE u+14,160+
d: POKE u+15,y2: REM
               ZASADNICZA PETLA

1160 FOR n=1 TO 2*d: LET f=USR 60435: GO SUB 930: POKE 60503
,PEEK 60503+1: LET x1=PEEK 60503: IF r1>17 OR r2>17 THEN  GO
 TO 1180
1170 LET f=USR 60435: GO SUB 930
1180 POKE 60511,PEEK 60511-1: LET x2=PEEK 60511: GO SUB 700
1200 LET f=12*(n=1 AND v=2): PLOT s+f,j: LET i1=s+f: LET j1=
j: LET z=z+35/d: LET s=INT (z+.5): IF dM>0 THEN  LET j=INT (
95.5+(Max-Mw)*80/dM)
1210 DRAW z-i1,j-j1: IF INKEY$="P" THEN  BEEP .5,9: GO TO 13
00:    REM przerwanie - klawiszem P
1220 NEXT n: IF v=1 THEN  GO TO 1260
1240 GO SUB 750
1250 LET a=CODE INKEY$: LET r1=INT r1: LET r2=INT r2: LET x1
=31: LET x2=32: LET v=4: GO SUB 800: LET f=USR 60232: LET vv
=0: IF a=80 OR a=0 THEN  GO TO 1250
1255 GO SUB 880: LET v=3: GO SJB 800: GO TO pp
1260 GO SUB 750: FOR f=1 TO 3: DRAW  INVERSE 1;3,0: DRAW 1,0
: LET z=z+4: NEXT f: GO TO 1150
1300 IF v=1 THEN  GO TO 1250
1310 GO TO 1240
8020 FOR n=1 TO 5: PLOT  INVERSE 1;x+4*n,y: GO SUB 8e3+CODE
d$(n)
8025 NEXT n: RETURN
8045 DRAW  INVERSE 1;0,2: DRAW 3,0: RETURN
8046 DRAW  INVERSE 1;1,0: DRAW 1,0: RETURN
8048 DRAW 3,0: DRAW 0,4: DRAW -2,0: DRAW 0,-4: RETURN
8049 DRAW  INVERSE 1;3,-1: DRAW 0,5: DRAW -1,-1: RETURN
8050 DRAW  INVERSE 1;0,4: DRAW 3,0: DRAW 0,-2: DRAW -2,0: DR
AW 0,-2: DRAW 2,0: RETURN
8051 DRAW  INVERSE 1;0,4: DRAW 3,0: DRAW 0,-4: DRAW -2,0: DR
AW  INVERSE 1;0,1: DRAW 1,1: RETURN
8052 DRAW  INVERSE 1;1,5: DRAW 0,-3: DRAW 2,0: DRAW 0,-2: DR
AW 0,4: RETURN
8053 DRAW 3,0: DRAW 0,2: DRAW -2,0: DRAW 0,2: DRAW 2,0: RETU
RN
8054 DRAW  INVERSE 1;4,4: DRAW -3,0: DRAW 0,-4: DRAW 2,0: DR
AW 0,2: DRAW -2,0: RETURN
8055> DRAW  INVERSE 1;0,4: DRAW 3,0: DRAW 0,-4: RETURN
8056 DRAW 3,0: DRAW 0,4: DRAW -2,0: DRAW 0,-3: DRAW 1,1: RET
URN
8057 DRAW 3,0: DRAW 0,4: DRAW -2,0: DRAW 0,-2: DRAW 1,0: RET
URN
8076 DRAW 2,2: DRAW  INVERSE 1;-2,2: DRAW 0,-5: DRAW 2,0: RE
TURN
8100 REM ***********************************************
8105 CLEAR 47615: LET q=6e4: PRINT AT 10,2;"JEDNORAZOWO PRZY
GOTOWUJE DANE";AT 12,8; FLASH 1;" LICZY 19 MINUT "
8110 FOR m=0 TO 7: LET b=(m=0 OR m=3 OR m=4 OR m=7): LET c=(
m>0 AND m<5): LET d=(m=1 OR m=4 OR m=6 OR m=7): RESTORE 8140
8115 LET i=26: FOR n=q+m*i TO q+m*i+i-1: READ a: POKE n,a
8120 NEXT n: NEXT m
8130 FOR m=n TO n+35: READ a: POKE m,a: NEXT m: LET i=39
8140 DATA 42,79,236,237,91,76+b,236,123,150-16*c,79-8*b,35,1
22,134+16*d,71+8*b,205,170,34,8,62,154,132,103,8,205,236,34
8150 DATA 42,79,236,43,43,34,79,236,42,74,236,43,34,74,236,1
7,1,0,237,82,242,96,234,201,33,0,218,17,0,64,1,0,8,237,176,2
01
8160 FOR g=0 TO 3: RESTORE 8230: LET b=(g=0 OR g=2): LET c=(
g=1 OR g=2)
8170 FOR n=m+g*i TO m+g*i+i-1: READ a: POKE n,a
8180 NEXT n: NEXT g
8190 FOR m=n TO n+58: READ a: POKE m,a: NEXT m
8230 DATA 62,12*c,50,145,92,42,81+8*b,236,237,91,83+8*b,236,
237,75,87+8*b,236,205,240,235,237,91,85+8*b,236,62,12-12*c,5
0,145,92,42,81+8*b,236,237,75,87+8*b,236,205,240,235,201
8240 DATA 34,79,236,237,83,74,236,237,67,76,236,121,50,78,23
6,205,96,234,62,0,50,145,92,201,33,0,64,17,0,194,1,0,24,237,
176
8250 DATA 33,0,194,17,0,218,1,0,8,237,176,201,33,0,226,17,0,
218,1,0,8,237,176,201
8260 FOR n=q+460 TO q+486: READ a: POKE n,a: NEXT n
8270 DATA 33,0,186,17,0,0,1,24,8,62,64,25,44,43,54,0,35,16,2
48,9,6,8,61,200,195,55,236
8300 FOR n=50000 TO 51000: POKE n,0: NEXT n
8310 LET ww=60519: LET ee=60583: LET r0=0: LET z=0
8320 FOR m=1e-9 TO 30.8 STEP .25: IF INT m<>r0 THEN  LET r0=
INT m
8330 FOR a=0 TO PI/4 STEP PI/15/m: LET x=m*COS a: LET y=m*SI
N a
8340 LET f=50000+INT (x+.5)+35*INT (y+.5): IF PEEK f<>0 THEN
  GO TO 8370
8350 POKE ee,x: POKE ee+1,y: LET ee=ee+2: LET z=z+1: POKE f,
1
8360 POKE ww+2*r0,z-256*INT (z/256): POKE ww+2*r0+1,INT (z/2
56)
8370 NEXT a: NEXT m: BEEP 1,1: PRINT AT 10,1;"SKASUJ WIERSZE
 od 8100 do 8400";AT 12,8;"(od **** do ****)"
8380 PRINT AT 15,5;" I NAGRAJ - GO TO 8500 ": STOP
8400 REM ***********************************************
8500 CLEAR : SAVE "GW.ZACM." LINE 8888: PAUSE 50: POKE 23736
,181: SAVE "GW.ZACM.Bt"CODE 59999,1420: PRINT AT 10,5;" CZEK
A NA WERYFIKACJE "
8510 VERIFY "GW.ZACM.": VERIFY ""CODE : RUN 10
8888 CLEAR 47615: LOAD "GW.ZACM.Bt"CODE : RUN 10
9999 REM          PO WPISANIU PROGRAMU - NACISNIJ RUN
```

MIKROKOMPUTERY JEDNOUKŁADOWE

Gdy zajrzymy pod obudowę któregoś z popularnych mikrokomputerów, takiego jak np. ZX Spectrum, Commodore 64 czy też IBM PC zobaczymy tam kilkanaście lub kilkadziesiąt układów scalonych: mikroprocesor, pamięci RAM i ROM, układy wejścia/wyjścia oraz wiele układów specjalizowanych (głównie do obsługi grafiki i dźwięku). Dzięki takiej architekturze mikrokomputer dysponuje bogatymi możliwościami pozwalającymi na zastosowanie go do najrozmaitszych celów — począwszy od gier, poprzez programowanie przez użytkownika, aż do sterowania skomplikowanymi urządzeniami zewnętrznymi.

Kiedy mikrokomputery zaczęły wkraczać do rozlicznych dziedzin zastosowań przemysłowych okazało się, że niejednokrotnie ich możliwości przekraczały potrzeby. Były jednak przy tym zbyt duże i pochłaniały zbyt wiele energii elektrycznej. Z tych właśnie przyczyn narodziła się koncepcja mikrokomputerów jednoukładowych, tzn. takich, w których jednostka centralna (mikroprocesor), pamięć i układy wejścia/wyjścia zostały umieszczone na pojedynczym podłożu krzemowym. Pierwsze mikrokomputery jednoukładowe pojawiły się jeszcze w latach siedemdziesiątych. Były to urządzenia stosunkowo prymitywne, których dobrym przykładem może być Motorola 14 1000 i MC 14 1200 z 4-bitową jednostką centralną-logiczną (ALU-Arithmetic-Logic Unit). Zawierały one pamięć programu (odpowiednik ROM) o pojemności 1 K×8 bitów oraz pamięć danych o pojemności 64 K×4 bity. Szybkość działania również nie była zbyt rewelacyjna: zegar dawał przebieg o częstotliwości 600 kHz.

Mimo tak poważnych ograniczeń układy te znalazły szerokie zastosowanie w kalkulatorach, zabawkach elektronicznych, kasach sklepowych, urządzeniach elektronicznych samochodów, systemach bezpieczeństwa, a także „inteligentnych" klimatyzatorach.

Sukces pierwszych mikrokomputerów jednoukładowych spowodował powstanie nowych, znacznie doskonalszych konstrukcji. Jedną z bardziej znanych jest rodzina MCS-48 firmy Intel wykonana w technologii HMOS. Część arytmetyczna należących do niej mikrokomputerów jest podobna: składa się z 8-bitowej jednostki ALU, akumulatora, przerzutnika wskaźników i dekodera instrukcji. Różne wersje wyposażone są w inne pamięci programu: np. 8748H i 8749H w pamięci EPROM (programowalna przez użytkownika), 8049AH i 8050AH w ROM (programowana w fazie produkcji), a wersje 8035AHL, 8039AHL i 8040AHL w ogóle nie są wyposażone w wewnętrzne pamięci programu. W zależności od wersji pamięć programu może mieć pojemność 1,2 K lub 4 K×8 bitów. Jednak niezależnie od wersji wszystkie mikrokomputery z rodziny MCS-48 mają identyczne kody poszczególnych operacji, a więc możliwa jest wymienność oprogramowania. Pamięć danych (RAM) ma pojemność 64, 128 lub 256×8 bitów dla wersji odpowiednio: 8048AH, 8049AH i 8050AH. W układzie 8048AW wbudowano 2 porty wejścia/wyjścia i 1 port wejścia umożliwiający wykonywanie rozgałęzień programu poprzez testowanie stanów wejść tego portu. Jako port można również wykorzystać szynę danych, która wyprowadzona jest na zewnątrz w celu podłączenia dodatkowej pamięci lub układów wejścia/wyjścia. Mikrokomputery jednoukładowe z serii MCS-48 mogą pracować z częstotliwością zegara wynoszącą 4, 6, 8 i 11 MHz. Cenną cechą układów tej rodziny jest możliwość wykonywania programu instrukcja po instrukcji (poprzez naciskanie klawisza), wykorzystywana podczas testowania programów.

W wersjach 8048AH, 8049AH i 8050AH istnieje możliwość odłączenia mikrokomputera od zasilania przy jednoczesnym pozostawieniu pod napięciem pamięci RAM. Dzięki temu dane są podtrzymywane przy poborze mocy wynoszącym 10—15 procent wartości nominalnej. Przewidując zasilanie układów MCS-48 z baterii projektanci zawarli w liście rozkazów specjalną instrukcję IDLE, która „zamraża" stan jednostki centralnej i zegara przyczyniając się do zmniejszenia poboru prądu ze źródła zasilającego. W takim stanie mikroprocesor może trwać aż do otrzymania sygnału przerwania lub sygnału RESET.

Mikrokomputery jednoukładowe Intela mają specjalną opcję przesłaniania pamięci wewnętrznej przez pamięć zewnętrzną, w której znaleźć się może np. procedura testująca. Warto może jeszcze powiedzieć, że kasowanie pamięci EPROM odbywa się przez naświetlanie ultrafioletem (fale o długościach krótszych niż 400 nm). Leżąca w pokoju pamięć tego typu traci zawartość po 3 latach, natomiast wystawiona na działanie bezpośredniego światła słonecznego już po tygodniu. Oczywiście, aby wprowadzić przekłamania wystarczą czasy znacznie krótsze. Chociaż mikrokomputery jednoukładowe pracują z reguły w zamkniętych obudowach, to wersje z pamięcią EPROM mają okienko do kasowania, zaklejane czarną, nieprzezroczystą naklejką.

Firma Intel oferuje dla swojej rodziny mikrokomputerów jednoukładowych liczne układy dodatkowe: pamięci, układy wejścia/wyjścia, sterowniki klawiatur, wyświetlacze i inne. Można z nich zbudować obwody odciążające jednostkę centralną w komputerach wyższej klasy, a jednocześnie nie przyczyniające się do radykalnego wzrostu ich gabarytów. Istnieje również „energooszczędna" wersja rodziny MCS-48 wykonana w technologii CHMOS. Normalnie taki mikrokomputer pobiera 60 mW mocy, podczas wykonywania rozkazu IDLE 25 mW, a po odłączeniu zasilania od jednostki centralnej (RAM nadal zasilana) zaledwie 10 mW. Podstawowym zastosowaniem tych układów są urządzenia zasilane z baterii. Tam, gdzie warunki środowiskowe znacznie odbiegają od standardów układów komercjalnej elektroniki (temperatura 0—70°C), a jednocześnie nie osiągają standardów wojskowych firma Intel proponuje mikrokomputery jednoukładowe z rodziny EXPRESS. Mogą one pracować w temperaturach −40°C— +80°C, a w temperaturze 125°C powinny funkcjonować poprawnie przez 7 dni.

Należy się spodziewać, że w najbliższych latach wzrastać będzie liczba urządzeń powszechnego użytku standardowo wyposażanych w układy elektroniczne kontrolno-sterujące bazujące na mikrokomputerach jednoukładowych. Wystarczy wymienić tu nowoczesne zestawy stereofoniczne, magnetowidy, programatory pralek automatycznych, kuchnie elektryczne, kalkulatory programowalne, modele kolejek elektrycznych a nawet mówiące lalki i misie. Równolegle ta kategoria mikrokomputerów znajduje zastosowanie w większych komputerach — przykładem „inteligentna" klawiatura IBM PC.

Krzysztof Zięcina
Jacek Nowicki

„Młody Technik — InforMik" wydaje Instytut Wydawniczy „Nasza Księgarnia"

Rada Redakcyjna: doc. dr Zygmunt Dąbrowski, inż. Jerzy Jasiuk, dr Zygmunt Kalisz, mgr Zbigniew Słowiński, mgr inż. Jerzy Siek, dr Zbigniew Płochocki, Piotr Postawka, mgr inż. Roland Wacławek, prof. dr hab. Andrzej K. Wróblewski (przewodniczący), mgr inż. Grzegorz Zalot.

Zespół redakcyjny: *„InforMik"* redaguje zespół *„Młodego Technika"*. Jerzy Klawiński (red. odpowiedzialny), Jacek Nowicki (red.), Lidia Sadowska-Szlaga (korekta), Józef Trzionka (redaktor naczelny), Roland Wacławek (software), Grzegorz Zalot (hardware), Izabela Żur (red. tech.).

Stali współpracownicy: Wojciech Apel, Tadeusz Basista, Jacek Jędrzejowski, Piotr Postawka, Marek Szczepański, Krzysztof Wiśniewski.

Adres redakcji: ul. Spasowskiego 4, 00-389 Warszawa, lub skr. poczt. 380, 00-950 Warszawa. **Telefony:** centrala: 26-24-31 do 36. Dział Łączności z Czytelnikami — wewn. 60, pozostałe działy: wewn. 42 i 47. Redaktor naczelny: 26-26-27 lub wewn. 87.

Warunki prenumeraty: ogólnie obowiązujące w kraju. **W STAŁEJ SPRZEDAŻY INFORMIK JEST W SALONIE WYDAWNICZYM „NASZEJ KSIĘGARNI"** ul. Spasowskiego 4 A.

Druk: Zakłady Graficzne w Katowicach. Zam. 1128/1333/7
Nakład 100 315 egz. K-93

MIKROKOMPUTER JEDNOUKŁADOWY FIRMY INTEL

Widoczny w powiększeniu obraz krzemowej płytki mikrokomputera jednoukładowego przypomina na pierwszy rzut oka lotnicze zdjęcie śródmiejskiej dzielnicy wielkiego miasta. Na otoczonej złotymi drucikami doprowadzeń powierzchni znajduje się 30 tysięcy tranzystorów. Obszar oznaczony cyfrą **1** to kasowalna pamięć programu — EPROM. Po lewej **(2)** blok pamięci RAM. W pobliżu układów pamięci mieszczą się układy adresowe oznaczone przez **3** i **4**. Największy obszar zajmuje dekoder instrukcji **5** rozbijający polecenia zawarte w programie na elementarne kroki rozwiązywane przez jednostkę arytmetyczno-logiczną **(6)**. W lewym dolnym rogu płytki znajduje się układ przerwań **(7)**, zaś w prawym górnym — układ wejścia/wyjścia spełniający rolę „centrali telefonicznej" do łączności mikrokomputera ze światem zewnętrznym.

Cena: 90 zł

Indeks nr 366013

